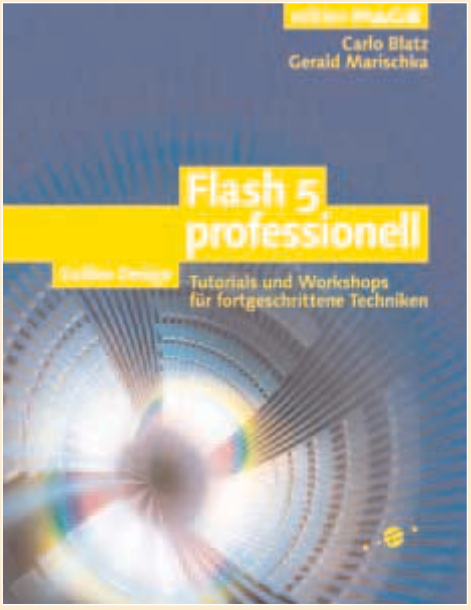


ISBN 3-934358-80-2

Dieses Buch liegt vollständig als PDF vor

done by
AnyBody

Flash 5 und ActionScript professionell

von Carlo Blatz und Gerald Marischka

Dieses Buch ist Freeware und nicht zum Verkauf bestimmt!

Carlo Blatz, Gerald Marischka

Flash 5 und ActionScript professionell

Tutorials und Workshops für fortgeschrittene Techniken

Galileo Design

Die Deutsche Bibliothek – CIP-Einheitsaufnahme
Ein Titeldatensatz für diese Publikation ist bei der
Deutschen Bibliothek erhältlich

ISBN 3-934358-80-2

© Galileo Press GmbH, Bonn 2001
1. Korrigierter Nachdruck 2001

Der Name Galileo Press geht auf den italienischen Mathematiker und Philosophen Galileo Galilei (1564–1642) zurück. Er gilt als Gründungsfigur der neuzeitlichen Wissenschaft und wurde berühmt als Verfechter des modernen, heliozentrischen Weltbilds. Legendär ist sein Ausspruch **Eppur se muove** (Und sie bewegt sich doch). Das Emblem von Galileo Press ist der Jupiter, umkreist von den vier Galileischen Monden. Galilei entdeckte die nach ihm benannten Monde 1610.

Lektorat Ruth Wasserscheid, Bonn
Korrektorat Sandra Gottmann, Bonn
Einbandgestaltung Helmut Kraus, Düsseldorf
Herstellung Petra Strauch, Bonn
Satz mediaService, Siegen
Druck und Bindung Bercker Graphischer Betrieb, Kevelaer

Das vorliegende Werk ist in all seinen Teilen urheberrechtlich geschützt. Alle Rechte vorbehalten, insbesondere das Recht der Übersetzung, des Vortrags, der Reproduktion, der Vervielfältigung auf fotomechanischem oder anderen Wegen und der Speicherung in elektronischen Medien.
Ungeachtet der Sorgfalt, die auf die Erstellung von Text, Abbildungen und Programmen verwendet wurde, können weder Verlag noch Autor, Herausgeber oder Übersetzer für mögliche Fehler und deren Folgen eine juristische Verantwortung oder irgendeine Haftung übernehmen.
Die in diesem Werk wiedergegebenen Gebrauchsnamen, Handelsnamen, Warenbezeichnungen usw. können auch ohne besondere Kennzeichnung Marken sein und als solche den gesetzlichen Bestimmungen unterliegen.

Inhalt

8 Einleitung
684 Index

12 Organisation

14 Vorbereitungen
20 Projektmanagement
34 Optimierung

40 ActionScript

42 Basics
46 Grundlagen von ActionScript
62 Objektorientierte Programmierung in Flash
80 Bézierkurven in Flash
86 Beispiele für den Einsatz von ActionScript
98 Smart-Filmsequenzen

108 Spieleprogrammierung mit ActionScript

110 Grundlagen
126 Flash-Huhn, Slotmaschine und Co.
134 Kollisionen, Highscores u.a.
146 Coco-Catch

190 Gestaltung

192 Grafik
198 Flash-Typografie
224 Animation mit Typografie
236 Sound
264 3D

288 HowTo
290 Typografischer Sitecheck
294 Sitecheck

324 Flash im Zusammenspiel
326 Zusammenarbeit mit anderen Programmen
336 Flash und RealVideo
362 Flash und QuickTime
370 Wiedergabe von Videosequenzen in Flash
374 Flash und JavaScript
396 Flash 5 JavaScript-Integration
422 Chatrooms in Flash
426 Erweiterung eines Flash-Movies mit dem Kommandozeileninterpreter
430 Weitere nützliche Tools für die Arbeit mit Flash

436 Server, Dynamik und Flash
438 Macromedia Generator
462 Swift-Generator
486 Anwendungsbeispiele für serverseitige Skripts
516 Flash und XML
524 Flash-Newsletter
528 Ming – SWFs ohne Flash
530 Einlesen der Uhrzeit und des Datums

580 Workshops und Anwendungsbeispiele
582 Effekte mit duplicateMovieClip
586 Passwortabfragen
604 Flash 5 Detection
610 Preloader
618 Flash Print
622 Cookies anlegen und auslesen
626 Mausposition und -verfolgung in Flash 4 und 5
640 Dynamisches Menüsystem erstellen

644 Anhang
646 FAQ
658 Wishlist Flash 6
660 Compiler-Fehlermeldungen
664 Webstatistiken
670 HTML 3.2-ASCII-Zeichen
678 Die Autoren dieses Buches
682 Die CD-ROM

Einleitung

Warum wer was wie für wen schreibt

Willkommen beim Flash 5-Profibuch. An diesem Buch haben mehrere Autoren mitgeschrieben, alle jeweils Spezialisten für einen Flash-Schwerpunkt. Das erfordert eine besondere Struktur und Hintergrundinformationen, die wir Ihnen vor dem Lernen mit auf den Weg geben möchten.

Zum Buch

Erst einmal herzlichen Glückwunsch zum Kauf unseres Buches!

Wie der Titel schon sagt, handelt es sich um ein Profibuch. Das bedeutet für uns eine Gratwanderung. Zum einen brauchen wir nicht so weit zurückzugreifen, zum anderen müssen wir aber auch vollständig bleiben. Sie werden also die eine oder andere Passage finden, die Ihnen bereits bekannt war, aber mit Sicherheit auch viel Neues.

Flash hat sich in seinen Wurzeln verändert. Das Programm ist von einer designerorientierten Umgebung zu einer programmierorientierten Umgebung mutiert. Es ist nicht mehr ausreichend, über das Programm selbst zu berichten, es muss tiefer in die Materie eingedrungen werden, um das »Werkzeug Flash« wirklich für individuelle Bedürfnisse nutzen zu können.

Dieses Buch soll Ihnen Aufschluss über die Grenzen der professionellen Möglichkeiten von Flash 5 bringen. Ziel ist es nicht, Grundbegriffe zu erläutern.

Es ist auch kein Buch, in dem es nur um ActionScript geht. Wir haben versucht, für Sie eine umfassende Zusammenstellung zu schaffen, welche die Materie nicht nur streift. Das sind sowohl neue Techniken von Flash 5 als auch Informationen, die vor oder nach dem Flashen wichtig sind.

Um mehr Platz für die Inhalte zu haben, verwenden wir nicht so viele Screenshots, wie Sie es z. B. aus Anfängerbüchern gewohnt sind. Wir setzen voraus, dass jeder weiß, wie Flash aussieht.

Eine wichtige Information noch: Wir werden bewährte Flash 4-Lösungen nicht zwangsläufig gegen eine Flash 5-Lösung austauschen, nur weil wir ein Flash 5-Buch schreiben. Wenn wir der Meinung sind, dass die Ziele mit einem mit Flash 4 kompatiblen Code genauso gut oder besser zu erreichen sind, empfinden wir es als unsere Aufgabe, diesen zu zeigen, da eine Abwärtskompatibilität nie schaden kann.

Dieses Buch ist ein Profiwerkzeug, das wir selbst gerne als Hilfestellung zu Verfügung gehabt hätten.

Ein Tipp noch zum effektiven Lernen: Sie sollten beim Nachvollziehen der Anwendungsbeispiele die FLA-Dateien der CD öffnen. Zwar wird der Code auch im Buch beschrieben, viele Zusammenhänge kann man aber besser in der Datei begreifen. Nähere Informationen zur CD befinden sich im Anhang.

Bitte beachten Sie auch die Webseite zum Buch unter www.galileo-press.de, wo wir Ihnen regelmäßig Aktualisierungen und Ergänzungen zur Verfügung stellen werden.

Die Autoren

Anfang 2000 trat Galileo Press (www.galileo-press.de) – seines Zeichens Verleger des ersten und erfolgreichsten deutschen Flash 4-Buches von Sascha Wolter – an Carlo Blatz heran, ein Profibuch zu Flash zu schreiben. Im Gespräch mit Gerald Marischka, ebenfalls Mitbegründer von Flashworker.de, sind wir zum Schluss gekommen, dass es nahezu unmöglich ist, ein umfangreiches Profibuch alleine zu schreiben. So haben wir die Bedingung gestellt, für diverse Spezialgebiete Coautoren einladen zu dürfen.

Carlo Blatz

Ich habe Anfang 1996 www.gabocorp.com gesehen und musste unbedingt wissen, wie das geht. Mit der ersten Flash 3-Version fand ich dann mein Medium und wagte mein erstes Projekt, siehe www.multipass.de. 150 Stunden, 70 Szenen, 6 Alukugeln und 5 Zahnstocher sind dabei draufgegangen. Heute würde man das wohl in winzigen ActionScripts klären, aber damals hat es gereicht, um aus einem Hobby eine der ersten Flash-only-Agenturen Deutschlands aufzubauen – www.powerflasher.de. Hier drehen wir unsere Referenzen mit einem solchen Flash 5-Skript frei im

Raum. Inzwischen sitzen im Haus mehrere Grafiker, Flasher, Kontakter etc. Seitdem klingelt bei uns das rote Telefon, wenn es darum geht, mehr aus Flash zu holen. Kunden wie Elsa, Electronic Arts, aber auch Agenturen wie Springer & Jakoby übergeben uns ihre Flash-Wünsche. Seit 1997 halte ich außerdem diverse Flash-Schulungen und Einzeltrainings ab.

Kontakt: office@powerflasher.de

Gerald Marischka

Zunächst einmal – servus!

Vor ca. drei Jahren habe ich mich im Netz selbstständig gemacht – damit wurde ein Hobby zum Beruf. Vor fast zwei Jahren ist Christian Pilsl auf mich zugekommen und hat von Flash geschwärmt. Nach einer anfangs skeptischen Phase haben wir gemeinsam mit Gustav Assem (Screendesign) die erste österreichische Full-Flash-Page unter www.die-waescherei.com ins Leben gerufen (sie hat, obwohl schon seit Mitte 1999 online, bei @d:tech.Europe 2000 gleich zwei Mal Gold geerntet). Inzwischen zählen McKinsey, die Bawag, der Bertelsmann-Verlag und einige andere Größen des nationalen und internationalen Marktes zu meinen Kunden.

Diesen Erfolg brachte im Wesentlichen die Wahl der jeweils am besten geeigneten Technologie. Flash spielte eine besondere Rolle, weil es ein wirklich interaktives Design und kürzere Download-Zeiten möglich machte. Die engste Zusammenarbeit besteht derzeit mit Screendesigner Gustav Assem und seinem Designteam www.tuer3.com, die Umsetzung erfolgt inzwischen in Zusammenarbeit von internationalen Experten unter meiner Leitung.

Kontakt: office@profipage.at

Die anderen Autoren haben wir entsprechend ihrer Qualifikation für die jeweiligen Themen ausgewählt. Sie sind Flashworker, Mitarbeiter oder Freelancer unserer Agenturen, Autoren anderer Bücher oder andere Koryphäen der Szene. Wer welches Kapitel geschrieben hat, haben wir am Ende des Buches für Sie zusammengefasst.

Flashworker

Ein Wort in eigener Sache: In diesem Buch wird des Öfteren die Rede von »Flashworker« sein. Da wir wohl keine unerfahrenen Leser haben, werden die meisten www.flashworker.de schon kennen. Dennoch, oder eher umso mehr, möchten wir die Gelegenheit nutzen, das Projekt vorzustellen.

»Flashworker« ist 1999 in der offiziellen Macromedia Newsgroup »macromedia.general.germany« entstanden. Die Newsgroup gilt seit Jahren als Wohngemeinschaft der deutschsprachigen Flasher. Auch wenn es dort sicherlich um mehr als Flash geht, so ist die Anzahl der Flash-Interessierten doch sehr groß. Es galt sich zu organisieren. Viele Fragen wurden inzwischen das x-te Mal gestellt, und viele Ideen fanden kein Zuhause. So schlossen sich etwa zehn Flasher zusammen und verteilten hochgesteckte Aufgaben. Am 1.1.2000 war es dann soweit, das größte deutschsprachige Portal von und für Flashworker ging online. Damals mit 20 Fragen im FAQ, 10 Tutorials, 20 FLAs etc. Inzwischen ist Flashworker Offizielle Macromedia Usergroup (MMUG) und zu *der* Seite für alle Flasher geworden, vom Newbie bis hin zum Profi schätzen alle die geballte Ladung an Informationen auf der Seite und im monatlich erscheinenden Newsletter.

Der Gedanke von Flashworker ist es, Know-how kostenlos zur Verfügung zu stellen, damit anderen Flashern zu helfen und selbst zu lernen. Längst hat sich eine umfangreiche Szene gebildet, die verstanden hat, dass das Zurückhalten von Know-how nur eine sehr kurzfristige Strategie ist. Die Kontakte und nicht zuletzt »Ruhm und Ehr«, die jedem Helfer zuteil werden, sind mehr wert.

Viele der Autoren dieses Buches haben bereits ein Tutorial bei Flashworker.de geschrieben, und es lag daher nahe, sich am ersten Flash-Profibuch zu beteiligen. Wer also neue Effekte oder Technologien mit Flash erstellt, ist jederzeit willkommen, sie bei Flashworker.de zu publizieren und seine Arbeit den derzeit über 50.000 Usern im Monat vorzustellen.

Organisation

14 Vorbereitungen

14 Grundsätzliches
15 Bühnengröße
17 Struktur
18 Aufbau eines FLAs

20 Projektmanagement

21 Die Analysephase
24 Das Storyboard
25 Das Pflichtenheft
26 Das Angebot
26 Der Zeitplan
28 Das Team
29 Filtern von Informationen
30 Die erste Abnahme
31 Die Tests
32 Fertig, und was jetzt?

34 Optimierung

34 Tricks für kleine Dateigrößen
35 ShapeTweens optimieren
36 MotionTweens optimieren
37 OnionSkinMarker – der Zwiebelschaleneffekt
38 Shared Libraries

Vorbereitungen

Rechts und links von ActionScript

Gute Skripte zu programmieren ist eine Sache, optisch anzusprechen eine ganz andere. Wir möchten Ihnen zeigen, wie man schöne Effekte produziert, worauf man grundsätzlich achten sollte und was man bei sauberen Animationen berücksichtigen muss.

Grundsätzliches

▸ Carlo Blatz

Das Wichtigste ist, einen Plan zu haben. Ein roter Faden und ein Konzept in der Animation machen ein Projekt erst rund und ansprechend. Alles andere wirkt wie eine sinnlose Aneinanderreihung von Effekten. Zum anderen spart man unter Umständen viel Zeit. Überlegt man sich vorher nicht, welche Probleme auftreten können, welche Anforderungen an Flash gestellt werden, macht man am Ende nur Kompromisslösungen oder muss sogar neu anfangen.

Hat man also einen solchen Plan, z. B. in Form eines Storyboards, gilt es erst einmal, die Daten zu sichten: Sind alle Bilder da, sind alle Texte da, liegen die Logos als Vektoren vor, gibt es die richtigen Schriften? Man sollte sich jeden Step vom Kunden absegnen lassen. Wurden die Texte geliefert, soll der Kunde unterschreiben, dass sie genau so verwendet werden sollen. Wurden die Layouts entworfen, soll der Kunde unterschreiben, dass sie genau so umgesetzt werden sollen. Und wurde die erste Seite animiert, soll der Kunde unterschreiben, dass die anderen Seiten genau so adaptiert werden können. Verlangt der Kunde dann im Nachhinein Änderungen, kann man diese auch berechnen – und meist erübrigen sich dann die Änderungswünsche. Im Angebot bzw. Auftrag sollte natürlich ein Korrekturschritt vereinbart sein, wo der Kunde die Chance hat, eine Liste von Änderungen (außerhalb der bereits unterschriebenen Steps) anzumerken. Sind diese Korrekturen wunschgemäß ausgeführt, sind alle weiteren Korrekturwünsche wieder berechnungsfähig. Genauere Informationen hierzu haben wir für Sie im Kapitel »Projektmanagement« ab Seite 20 bereitgestellt.

Bühnengröße

Ist es nun endlich soweit, dass es an Flash geht, muss die schwierigste aller Fragen geklärt werden – die Bühnengröße. Bei Bannern und Pop-ups sowie festen Frames ist die Antwort einfach, wenn man aber für variable Frames oder gar Fullscreen-Browser entwickelt, sollte man für diese Frage mehr Zeit verwenden. Als Erstes könnte man sich z. B. den Screenshot von einem gängigen Browser in Fullscreen bei einer gängigen Auflösung anzeigen lassen und die nutzbare Fläche des Browsers ausmessen. Die derzeit gängigste Konfiguration ist PC, Windows, Internet Explorer 5 bei 1024 x 768 Pixel. Die nutzbare Fläche des Browsers entspricht hier in der Standardeinstellung der Symbolleisten ca. 1000 x 600 Pixel. Legt man die Bühne in dieser Größe an, hat man schon für eine maximale Menge von Benutzern einen guten Wert.

Leider gibt es auch andere Browser, Auflösungen, Symbolleisteneinstellungen und den Macintosh, der in dieser Form keinen standardmäßig genutzten Knopf für Fullscreen hat. Man muss sich also überlegen – eigentlich muss das schon der Grafiker –, was passieren soll, wenn der User eine andere Auflösung hat oder seinen Browser anders skaliert. Die Auflösung an sich ist ja nicht die eigentliche Schwierigkeit; dank Vektoren lässt sich Flash ja ohne Qualitätsverlust skalieren. Vielmehr sind die Proportionen interessant. Die sichtbare Fläche eines Browsers ist meistens im 4:3-Format. Doch bei präzisen Layouts kann das Verhältnis 4 zu 3,4 oder 4,1 zu 3 schon viel ausmachen.

Der Schlüssel liegt in den Exporteinstellungen. Hier kann man durch diverse Optionen beeinflussen, wie Flash mit dem Anschnitt umgehen soll. Publiziert man auf einer festen Pixelgröße, ist es nicht so wichtig; bei 100 % x 100 % können aber doch sehr unangenehme Effekte auftreten.

Bei www.sebo.de haben wir z. B. den Flash-Film mit 100 % x 100 % publiziert. Ausgerichtet ist er an der linken Seite auf der horizontalen Mitte. Angelegt wurde der Film auf 1000 x 600 Pixel. Verändert man nun aber die Proportionen, entsteht keine unschöne Kante, sondern bei horizontaler Stauchung ein breiter Rand und bei vertikaler Stauchung eine längere »Randlinie«. Es sind also durchaus 30 % Toleranz eingeplant, die dem Layout nicht schaden.

Auch für den Import der Pixelbilder (mehr dazu unter »Tipps zum Import von Pixelbildern« ab Seite 192) ist es wichtig, die genaue Bühnengröße zu wissen. Ohne Pixelbilder könnte man den Film zwar genauso in 500 x 300 Pixel anlegen, mit Pixelbildern würde die Skalierung aber eine schlechte Bildqualität bedeuten. Wird der Film auf 2000 x 1200 Pixel an-

Abbildung 1 ▶
Ideal

Abbildung 2 ▶
Horizontal gestaucht

gelegt, ist die Bildqualität zwar in allen Auflösungen am besten, die Dateigröße aber unsinnig groß. Es ist also am besten, für eine durchschnittliche Auflösung zu optimieren.

Sind diese Überlegungen getroffen und die Bühnengröße festgelegt, müssen die Daten sauber in Flash importiert werden (mehr dazu unter »Übernahme von Screendesigns« ab Seite 194).

Unter Umständen muss man sogar Einiges nachbauen. Masken aus FreeHand z. B. machen oft Probleme und können optimiert werden.

◂ Abbildung 3
Vertikal gestaucht

Sind alle Daten im Flash, kann man beginnen, sie auf die richtigen Ebenen zu verteilen und Objekte in Symbole umzuwandeln. Spätestens hier sollte klar sein, wie der spätere Film aufgebaut werden soll: ob alles auf einer Zeitleiste abläuft, ob es Szenen gibt, die man ordnen kann, ob Teile in Filmsequenzen oder sogar in externe SWFs gehören.

Struktur

Die Struktur ist ebenfalls eine sehr wichtige Komponente sowohl in der Arbeit, wie bereits angesprochen, als auch im späteren Ergebnis. Das In-

ternet ist ein sehr schnelles Medium. Durchschnittliche Verweilzeiten liegen im Sekundenbereich. Da ist es dringend notwendig, den User zu entlasten und ihm eine Struktur zu bieten, die er schnell durchschauen kann. Gut – nun bietet sich gerade Flash an, ausgefallene Sachen zu machen, die eben in kein Baumdiagramm einzuordnen sind, aber auch hier sollte man gewisse Regeln einhalten.

So sollte eine Seite zum Beispiel nicht zu tief verschachtelt sein. Maximal zwei Ebenen tief sollte man in eine Struktur tauchen können. Sind weitere Unterpunkte notwendig, kann man auch Pop-ups oder andere Abhebungen verwenden, die verhindern, dass sich der Anwender verläuft.

Man sollte die Navigation auch nicht innerhalb der Seite ändern. Eine Seite muss nicht aussehen, wie man es gewohnt ist, aber sie sollte auch nicht in sich widersprüchlich sein. Der Mensch ist ein Gewohnheitstier, und zum Glück gewöhnt er sich recht schnell – wenn man auf einer Seite aber gar kein System findet, strengt das unnötig an, und dafür bedankt sich niemand.

Zur Struktur gehört auch, dass man das Rad nicht auf jeder Seite neu erfindet. Man sollte sich auf einige Animationen oder Arten der Überblendung etc. festlegen und diese konsequent verfolgen. Nur so kann ein geschlossenes Konzept ohne Bruch entstehen.

Aufbau eines FLAs

Auch das hat etwas mit strukturiertem Arbeiten zu tun. Nicht nur, wenn man Projekte im Team entwickelt, ist es sinnvoll, sein FLA möglichst sauber und aufgeräumt zu halten. Zum einen kann man so schneller arbeiten, macht weniger Fehler und findet sich einfacher zurecht, zum anderen muss man die Datei eventuell anderen übergeben oder Monate später noch mit ihr arbeiten. Jeder hat wohl sein eigenes System, Ordnung zu schaffen. Daher möchte ich nur einige Möglichkeiten vorstellen.

Die **Ebenen** sollten nach einem bestimmten Prinzip angeordnet sein. Eine Reihenfolge ergibt sich durch die optische Notwendigkeit.

Was aber ist mit Sound, Aktionen und Ebenen, auf denen Elemente optisch nebeneinander angeordnet sind? Wir haben z. B. eingeführt – und das predige ich auch in meinen Schulungen: Befehle kommen von oben. Nichts ist lästiger, als auf 50 Ebenen nach einer Aktion zu suchen. Wenn man sich einfach oben eine Ebene anlegt, z. B. »Steuerungsebene« genannt, und festlegt, dass nur dort Aktionen der Zeitleiste platziert wer-

den, behält man hier schon einmal den Überblick. Hierher gehören auch Bezeichnungen, die mit Goto angesprungen werden.

Ähnlich machen wir es auch mit **Sound**. Die Soundspuren befinden sich direkt unter der Steuerungsebene und beinhalten ebenfalls keine Objekte der Bühne, sondern nur Sound. Ausnahmen gibt es bei animationssynchronen Event-Sounds, die zum entsprechenden Keyframe gelegt werden, um eine Zugehörigkeit besser zu kennzeichnen.

Unter den Soundebenen fangen dann erst die Ebenen mit Elementen an, die auf der Bühne liegen. Aber selbst hier gibt es noch System. Oben liegen, wenn möglich, die Ebenen, die dauerhaft zu sehen sind, wie z. B. die Navigation oder ein Logo. Dass der Hintergrund ganz unten liegt, brauche ich wohl nicht zu sagen! Diese Ebenen hier kann man auch eindeutig benennen. Eventuell ist es auch sinnvoll, jeweils Ebenen für die Headline, die Subline und den Text o.Ä. zu reservieren, wenn es da eine Regelmäßigkeit gibt. Durch dieses System reduzieren sich die eigentlichen Animationsebenen auf ein Minimum. Da ist es unbedingt notwendig, den Überblick zu behalten.

Mit den restlichen Ebenen sind wir möglichst sparsam, um nicht unnötig viele zu haben. Wenn eine Animation beendet und nicht mehr sichtbar ist, kann sie wieder für andere Animationen benutzt werden. Wenn auf der Bühne eine Filmsequenz liegt, in der sich wiederum eine komplexe Animation befindet, fügen wir diesen Keyframes kurze Kommentare hinzu, damit man sie schnell findet. Idealerweise kann man sich in einer Zeitleiste zurechtfinden, ohne die Bühne zu sehen.

Was die **Zeitleisten** selbst betrifft, muss man für sich die Entscheidung treffen, ob man Szenen verwendet oder sogar in externe Files ausweicht. Das kommt immer auf das jeweilige Projekt an.

In der **Bibliothek** kann es auch oft wüst zugehen. Bei 100 und mehr Symbolen ist Ordnung wichtiger denn je. Die Bezeichnungen sollten stets eindeutig sein und auch nicht bei vergleichbaren Symbolen variieren. So würde ein animierter Button mindestens aus einer Grafik (natürlich mit Symboleigenschaft Filmsequenz – siehe »Optimierung« ab Seite 34), einer Filmsequenz und dem eigentlichen Button bestehen.

Ordner können in der Bibliothek nochmals aufräumen. Hier muss sich aber wieder jeder sein eigenes System schaffen, das er auch nach Monaten noch nachvollziehen kann. Manche ordnen die Symbole entsprechend der Szenen in Ordner, andere haben einen Ordner für Filmsequenzen, einen anderen für Buttons etc. Das richtige System ist das, das alle am Projekt beteiligten verstehen.

Projektmanagement

Oder: Ist Freizeit Luxus?

Sonntag, 4 Uhr früh, und noch immer keine Zeit zum Schlafen, der Aschenbecher voll und noch 50 Mails unbeantwortet. Genug Kunden, genug Geld, aber keine Zeit, es auszugeben. Der Kunde zittert, der Soundtechniker ist krank, der Provider scheinbar verstorben – Abgabe ist morgen. Wie managt man den »ganz normalen« Alltag?

▸ Gerald Marischka

Allgemeines

Ein multimediales Projekt, sofern es eine bestimmte Grösse übersteigt, ist mit Arbeit verbunden. Arbeit will geplant sein. Mit einer guten Planung steht oder fällt ein Projekt. Dies gilt nicht nur bei großen Projekten, die Grundsätze müssen auch bei kleinen und kleinsten Projekten eingehalten werden.

Webagenturen verfügen über eigene Verkäufer, Projektleiter, Screendesigner, Programmierer, Musiker oder Soundspezialisten, über Datenbank-Gurus, Werbetexter und weitere Spezialisten.

Alle wissen selbstverständlich, was zu tun ist, keiner hat Fragen, und alles ist ganz leicht. Auftrag, Arbeit, Rechnung, fertig. Soweit zur Theorie.

In der Praxis überschneiden sich Aufgaben, es werden projektabhängig Spezialisten hinzugezogen, man arbeitet meist an mehreren Projekten zur selben Zeit. Alle Beteiligten sind, Gott sei Dank, Menschen und machen daher Fehler. Es kommt zu Missverständnissen, Krankheiten und persönlichen Prioritäten.

Das Werkzeug zeigt einmal wieder seine schwache Seite, und die Daten sind verloren, weil ausgerechnet das letzte Backup auch Fehler zeigt (ist doch immer so, oder?).

Falls Ihnen das nur hin und wieder, in Spitzenphasen, zu schaffen macht, ist es weiter nicht tragisch. Wenn es allerdings so ist, dass der beschriebene Zustand eher mit der Formulierung »es wird schon besser«

abgetan wird, sollte man sich Gedanken darüber machen, denn es wird nicht besser!

Es muss eine Vorgehensweise festgelegt werden. Was macht wer wie und bis wann?

Die Analysephase

Ein Kunde zeigt Interesse, man trifft sich und redet über das zukünftige gemeinsame Projekt.

Jetzt gilt es, die Nerven zu bewahren und auch bei sehr entschlossenen Kunden professionell zu bleiben. Gerade ein begründetes Nein ist meist professioneller als ein fortwährendes Akzeptieren.

Wenn etwas nicht realisierbar ist, dann sagen Sie das Ihrem Kunden, auch wenn er begeistert von seinen Ideen ist. Es ist Ihr Job, den Kunden objektiv zu beraten, damit seine Ziele umgesetzen werden können.

Auf Kundenseite erwarten Sie Verwaltungsräte, Geschäftsführer, Verkäufer und Marketingleute und oft auch die interne EDV-Abteilung. Alle haben eine Meinung, alle Meinungen sind verschieden. Alle diese Meinungen unter einen Hut zu bringen ist die Aufgabe eines Projektleiters. Viel Diplomatie und Fingerspitzengefühl ist da gefragt, um einen Auftritt für alle Beteiligten erfolgreich abzuschließen.

Die oft benutzte und billige Ausrede »der Kunde wollte es so« streichen Sie bitte in Zukunft aus Ihrem Wortschatz, denn schlechte Qualität ins Netz zu stellen kostet alle Beteiligten nur unnötig Zeit und Geld.

Die Analysephase hilft Ihnen, Missverständnisse und logische Fehler schon im Keim zu ersticken. Auch sieht der Kunde selbst, welche seiner Ideen gut und welche zu verbessern sind. Die Ziele des Kunden zu verstehen ist leichter als man denkt.

Wir unterscheiden zwischen drei Analysephasen:

- Anforderungsanalyse
- Benutzeranalyse
- Task-Analyse

In der **Anforderungsanalyse** müssen Entscheidungen darüber getroffen werden, welche Informationen präsentiert werden sollen und zu welchem Zweck.

Es sind Fragen zu beantworten wie:

- Was ist der eigentliche Zweck des Internetauftritts?
- Welche Zielsetzungen werden damit verfolgt?
- Welche Information ist daher für den Benutzer wesentlich und soll von ihm besonders rasch gefunden werden?

Eine der wohl wichtigsten Fragen hier ist auch die nach dem **Budget**. Welches Budget ist für die Verwirklichung der Anforderungen vorgesehen?

Wenn Sie nicht wissen, in welchen Regionen Sie sich bewegen, laufen Sie Gefahr, einen Ferrari zu planen, obwohl nur Budget für einen Mercedes vorhanden ist. Auch sehen Sie schon jetzt, ob das Budget realistisch ist, und damit auch, ob das Projekt erfolgreich werden kann. Bestehen Sie zumindest auf eine konkrete Aussage – alles andere ist Zeitverschwendung.

In der **Benutzeranalyse** wird ein Modell des künftigen Benutzers entworfen. Dazu können allgemeine Untersuchungen über Internetbenutzer herangezogen werden.

Weiter sind Überlegungen anzustellen, an welche Internetbenutzer sich das System richtet. In welchem Verhältnis stehen diese Benutzer zur Organisation? Sind es Internetbenutzer, die zufällig vorbeisurfen? Welche Sprache sprechen sie? Sind es Touristen? Was kann man ihnen bieten?

In der **Task-Analyse** wird untersucht, welches Ziel ein Benutzer verfolgen wird bzw. welche Aufgaben er/sie lösen möchte. Die Antworten ergeben sich aus den Ergebnissen der vorangegangenen Anforderungs- und Benutzeranalyse. Falls es hier also zu scheinbar nicht lösbaren Fragen kommt, so liegt das daran, dass Sie zuvor einfach nicht genug Fragen gestellt oder Antworten bekommen haben.

Ein Beispiel

Der Kunde ist ein amerikanischer Spraydosenhersteller.

In der Anforderungsanalyse wird festgestellt:

- Der Zweck des Internetauftrittes ist die Erweiterung des Marktes.
- Das Ziel ist es, Jugendliche zwischen 18 und 28 Jahren dauerhaft zu binden.
- Die dazu nötigen Informationen des Benutzers sind seine Hobbys, um individuell auf ihn eingehen zu können.

In der Benutzeranalyse wird es jetzt genauer, denn wir wissen bereits, dass wir auf 18- bis 28-Jährige zielen. Es handelt sich nicht um Mitarbeiter der Organisation.

Bei der Sprachwahl kommt es zu einem längeren Gespräch. Da nur innerhalb der USA geliefert werden soll, liegt es auf der Hand, die Homepage in englischer Sprache zu entwickeln. Allerdings, wie viele Sprayer sind neue Staatsbürger? Welche Sprache sprechen diese? Gibt es eine szeneninterne Sprache (Dialekt)? ... aktuelle Daten müssen eingeholt werden, um eine objektive Entscheidung treffen zu können. Was ist innerhalb der Szene in und was out?

Die Antworten auf diese Fragen geben uns auch gleich Antwort auf die Frage: »Was kann man bieten?«

In unserem Beispiel stellen wir fest, dass ein ganz spezielles Spraymuster absolut in ist. Da dieses Spraymuster mit einem einfachen Aufsatz auf die Spraydose möglich und darüber hinaus in der Fertigung sehr günstig ist, entschließt sich der Kunde, diesen Aufsatz jedem Besucher der Homepage zu schenken. Um den Benutzer dauerhaft zu binden, soll es wöchentliche Sonderangebote und monatliche Preisausschreiben geben. Die Sonderangebote werden Kombinationen aus bereits bestehenden Angeboten sein. Die Preisausschreiben müssen verschiedene Kategorien abdecken, weil die Motivation innerhalb der Zielgruppe unterschiedlich ist (Prominenz, Gestaltung, Aussage, Technik ...).

Die Task-Analyse fällt uns nun leicht. Wir haben Daten über die Zielgruppe gesammelt, wir wissen, was in ist, wir sprechen die gleiche Sprache, und wir haben auch abseits der interessant gestalteten Homepage einen Motivationsgrund, um die Seite zu besuchen. All die Dinge, die wir als besonders wichtig herausgearbeitet haben, sollen selbstverständlich besonders leicht und schnell gefunden werden. Eine Menüstruktur wird festgelegt, und Inhalte der Seiten werden schon einmal besprochen.

Sie fragen sich nun, warum Sie diese Fragen klären müssen. Es könnte der Kunde doch auch Ihre Ideen verwenden und mit jemand anderem umsetzen ...

Diese Vorgehensweise ist sinnvoll, da

- Ihr Kunde oft noch keine Erfahrung mit neuen Medien hat und daher nicht von selbst ein gutes Konzept erarbeiten kann.
- Ihr Kunde denkt, das Netz löst all seine Probleme. Er hat daher Wünsche, die Sie als Profi nun relativieren und dann umsetzen müssen.
- Sie sonst kein faires Angebot erstellen können, weil Sie nicht wissen, was zu tun ist.
- sich der Kunde besser jetzt für Sie als in zwei Wochen für jemand anderen entscheidet. Falls er Ihre Ideen weiterverwendet – Pech gehabt (abgesehen davon kann man auch für die Analysephase Geld verlangen).

- Sie zwar ohne diese Analysephase irgendeine Seite erstellen könnten, aber diese Zeiten vorbei sind– Sie sind jetzt Profi.
- sonst weder eine gute Umsetzung noch eine gute Referenz möglich ist. Ihre Referenz ist immerhin Ihre beste und günstigste Werbung.

Das Storyboard

Die Analyse ist beendet, der Kunde möchte einen Designvorschlag sehen.

Basierend auf der Analysephase erfolgt ein erstes Design. Das erste Layout bezieht sich vor allem auf die Struktur und »Verlinkung« der Homepage.
Es werden zunächst nur Entscheidungen über den grundsätzlichen Aufbau der Seiten getroffen. Es wird definiert, wo die eigentliche Information auf der Seite zu finden sein wird, wo Logos, Links, E-Mail-Anfragemöglichkeiten, Navigationshilfen etc. positioniert sein sollen.

Ich habe inzwischen x verschiedene Definitionen und Varianten von Storyboards gehört/gesehen. Meist ist es nicht wirklich vom Kunden, sondern von Ihrem internen Ablauf abhängig, wie Sie nun weiter vorgehen werden.

Fakt ist, das Ziel eines Storyboards ist nicht, eine Seite zu programmieren – das ist Zeitverschwendung.

Ziel ist es, dem Kunden die Ideen und Umsetzungsvorschläge des Teams näher zu bringen. Ob Sie das anhand einer Bleistiftskizze oder einer Bildschirmpräsentation tun, ist reine Geschmacksfrage.

Ein Storyboard ist Optik – nicht technische Umsetzung. Gehen Sie noch nicht zu sehr in die Technik, das schränkt Sie nur ein. Falls aus technischen Gründen eine Adaption des Storyboards (nach der Abnahme durch den Kunden) nötig werden sollte, müssen Sie nachträglich eine akzeptable Lösung finden (machen Sie Ihren Kunden bei der Präsentation auf diesen Faktor aufmerksam!). Nur so können Sie wirklich herausragende Seiten gestalten.

Persönlich rate ich Ihnen, eine Bildschirmpräsentation zu machen. Die Farben und Größen stimmen, und man kann die Seite in der zukünftigen Umgebung betrachten.

Das Pflichtenheft

Der Kunde begeistert, das Storyboard abgenommen, alle sind euphorisch – jetzt heißt es Vorsicht, denn sonst kommt es zu Missverständnissen.

Am Anfang einer jeden Webkarriere steht das Wort und/oder der Handschlag. Bald merkt man allerdings, dass sich zwar alle einig waren, es aber auf Grund von Missverständnissen zu Fehlinterpretationen gekommen ist. Der Kunde ist nicht zufrieden, das Team macht die x-te Änderung und bewegt sich im Kreis. Kosten/Nutzen klaffen auseinander.

Das Pflichtenheft, wie ich es hier vorstelle, soll diesem Szenario entgegenwirken, indem es Missverständnisse ausschließt und die interne Organisation erleichtert. Es ist die Basis für ein Angebot.

Kernaussagen eines Pflichtenhefts:

- Was soll geschehen (z. B. Homepage für eine bestimmte Zielgruppe mit speziellen Inhalten usw.)?
- Mit welchen Mitteln sollen die Ziele umgesetzt werden (Storyboard, Kompatibilität und Technik allgemein)?
- Was ist Teil des Auftrages (z. B. Projektumsetzung, Animationen, Design)?
- Was ist ausdrücklich nicht Teil des Auftrages (z. B. Anpassung der bestehenden Datenbank, Eintragung in Suchmaschinen)?
- Bis wann macht wer was (grober Zeitplan – Analysephase, Design, Abnahme, Umsetzung, Fertigstellung)?

Kommt es nachträglich zu Veränderungen der Aufgabenstellung, so muss ein neues Pflichtenheft und ein zusätzliches Angebot erstellt werden. Die Akzeptanz der Kunden ist hier sehr groß, denn wenn man einen roten Mercedes bestellt, dann bekommt man auch einen roten Mercedes geliefert – soll es nun doch ein goldener Ferrari werden, hat das natürlich Folgen.

Ziel eines Pflichtenheftes ist es nicht, nett zu sein – hier geht es um Fakten.

Nach der ersten Abgabe des Pflichtenheftes kommt es meist noch zu Anpassungen, ergänzenden Details und anderen Änderungen. Freuen Sie sich darüber, all die Dinge, die jetzt geklärt werden, hätten später weit mehr Zeit, Geld und Nerven gekostet. (Versionsnummer nicht vergessen!!!)

Koordinaten festhalten
Halten Sie im Anhang des Pflichtenheftes auch die Koordinaten aller Beteiligten fest, das erleichtert Ihnen später nicht nur die Suche, sondern ermöglicht darüber hinaus auch eine rasche, flache Kommunikation.

Das Angebot

Das Pflichtenheft ist abgenommen? Ein Angebot dürfte nun kein Problem sein.

Sie wissen, was zu tun ist, Sie kennen Ihren Preis, und Sie wissen bereits aus der Anforderungsanalyse, welches Budget grob zur Verfügung steht. »Wir beziehen uns auf das Pflichtenheft XY in der Version Z vom 01.01.2001. Unser Angebot lautet: ...«

An sich muss dazu nicht viel mehr gesagt werden, denn Sie sind ja bereits Profi. Eine Sache liegt mir allerdings diesbezüglich am Herzen, und ich möchte ihr daher noch einige Zeilen widmen.

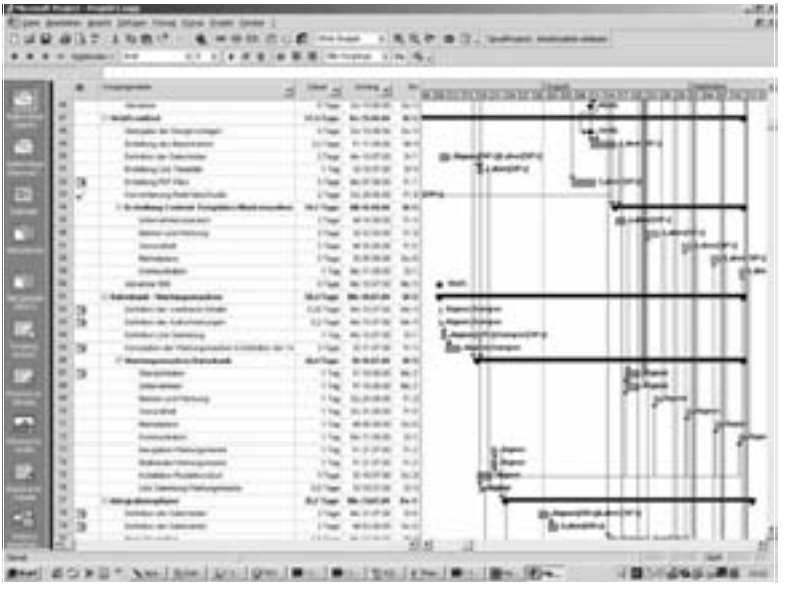
Es kommt mir immer öfter zu Ohren, dass angebliche Profis nicht wissen, was man berechnen kann. Auch bekommen manche Leute Probleme, wenn sie ihr eigenes Angebot durchlesen, und fürchten, dem Kunden erscheine der Preis viel zu hoch.

Bedenken Sie: Ein Angebot ist nicht mehr und nicht weniger als ein Angebot. Sie werden sicher nicht nur wegen einem zu niedrigen oder zu hohen Angebot aus dem Rennen geworfen. Über ein Angebot kann man reden – ein Angebot ist erst dann das letzte Wort, wenn es angenommen wurde. Bevor Sie unter dem vorgesehenen Budget des Kunden liegen, positionieren Sie sich lieber darüber – gespart werden muss meist so und so.

Der Zeitplan

Bis jetzt haben wir uns nur damit beschäftigt, den Auftrag zu bekommen. Alles steht auf Go!, jetzt muss ein genauer Zeitplan her, denn der bereits im Pflichtenheft erstellte enthält nur die wichtigsten Daten (Analysephase, Design, Abnahme, Umsetzung, Fertigstellung).

Ob man nun den »klassischen« Weg geht und einen Terminplaner mit dem Bleistift vollzeichnet oder z. B. Excel verwendet, bleibt dabei natür-



◀ **Abbildung 1**
Projektverwaltung mit Microsoft Project

lich jedem selbst überlassen. Der eleganteste Weg ist sicherlich, das Projekt mit Microsoft Project zu verwalten.

Wichtig ist nur, dass ein Zeitplan gemacht wird, sonst werden Terminverschiebungen rasch sehr unübersichtlich. Setzen Sie **Points of no Return,** um verschiedenen Phasen der Entwicklung einen wirklichen Endpunkt zu geben. Diese Punkte haben immer dann Sinn, wenn an einem bestimmten Punkt der Entwicklung Daten zusammenlaufen und/oder Daten übergeben werden müssen. Bedenken Sie Ausfälle durch Krankheit und andere unvorhersehbare Probleme. Setzen Sie vor Points of no Return immer einen realistischen Buffer von einigen Tagen, sonst fällt Ihnen der ganze Zeitplan wegen einem Teammitglied aus dem Rahmen, und das ist nicht akzeptabel.

Zeitpläne müssen immer in Zusammenarbeit mit dem Team umgesetzt werden. Bevor Sie sich Ihr Team zusammenstellen, müssen Sie allerdings schon grob wissen, wann Sie was brauchen, denn das wird eine der ersten Fragen jedes Teammitglieds werden.

Das Team

Das Team ist der Schlüssel zum Erfolg – das ist kein netter Spruch, sondern das Einmaleins.

Ob Sie Ihr Team innerhalb einer Firmenstruktur zur Verfügung haben oder auf ein Netzwerk von Freiberuflichen zugreifen, macht prinzipiell keinen Unterschied.

Persönlich rate ich dazu, auf ein Netzwerk zuzugreifen. Es ist fast immer zu teuer, ausreichend Experten auf allen Gebieten anzustellen. Beide Varianten haben Vor- und Nachteile, und bei beiden Varianten gelten ähnliche Regeln.

Wenn genügend Budget vorhanden ist, »gönnen« Sie dem Projekt die nötigen Spezialisten. Das kostet zwar sicher mehr Organisationszeit, ist aber langfristig günstiger und die einzige Möglichkeit, herausragende Projekte ins Netz zu bringen. Die technischen Möglichkeiten steigen ständig, und auch Flash teilt sich mehr und mehr in verschiedene Spezialgebiete.

Sie sind Profi, versuchen Sie dennoch, nicht überall den Experten zu mimen, denn ich kenne niemanden, der Experte auf allen Gebieten ist. Nutzen Sie die Möglichkeit, Experten zusammenzubringen und gemeinsam eine Vision zu verwirklichen. Falls Sie die Vorschläge Ihres Teams einmal nicht teilen – auch gut –, es liegt an Ihnen zu sagen, was Sie möchten. Holen Sie Meinungen ein, hinterfragen Sie Vor- und Nachteile, und treffen Sie eine Entscheidung.

Fair bleiben! Kunden werden sicherlich oft Druck auf Sie ausüben. Ob das nun zusätzliche Wünsche, ein neuer Preload oder die x-te Textänderung sind, geben Sie diesen Druck nicht weiter, sonst verliert der Letzte in der Kette – das ist nicht akzeptabel. Falls Sie Ihren Job gut gemacht haben, geht aus dem Pflichtenheft hervor, wer die Kosten zu tragen hat. Ein neuer Preload bedeutet auch Verschiebungen im Zeitplan. Machen Sie dem Kunden klar, welche Konsequenz sein zusätzlicher Wunsch hat. Das Team mit ständigen unentgeltlichen Erweiterungen und Änderungen zu »quälen« und dennoch auf den Zeitplan zu bestehen ist keine Lösung.

Haben Sie einen Fehler gemacht, so zahlen Sie auch dafür – möchte der Kunde Zusätzliches, so lassen Sie ihn dafür zahlen. Klingt simpel – ist simpel.

◀ **Abbildung 2**
Mailverwaltung mit Outlook Express

Filtern von Informationen

Ich habe lange überlegt, ob ich auf diesen Punkt eingehen soll, immerhin sind Sie ja Profi und brauchen keine Tipps, wie Sie Informationen filtern, oder doch?
Ein Blick in meine Mailbox verrät mir, dass auch so manchem Profi Tipps zu diesem Thema näher gebracht werden sollten.

Gewiss ist es nicht leicht, aus 200 bis 300 Mails, Newsgroup-Beiträgen, klassischen Medien und Anrufen die wichtigen Informationen zu filtern. Glauben Sie mir – ich kenne das Problem gut, und auch ich drohe oft in Informationen zu ersticken.

Um nicht zu sehr vom Thema abzuweichen, möchte ich hier nur auf das Filtern von Mails eingehen.

Tipps zum Umgang mit Mails

1. Richten Sie verschiedene Ordner in Ihrem Mailprogramm ein. Zumindest sollten Sie für Projekte (Kunde, Team ...), Privates, Tipps & Tricks, Altes und verschiedene Prioritäten Ordner haben.

2. Eine Mail, die Sie gelesen haben, sollte nicht in der Inbox bleiben. Auch wenn Sie sie nicht sofort beantworten möchten oder können, schieben Sie sie in einen Prioritäten-Ordner. Diesen können Sie dann abarbeiten.
3. Wenn Sie sich nicht sicher sind, ob Sie eine Mail beantworten sollen, und sie auch nicht zu einem Projekt gehört, schieben Sie sie in einen Altes-Ordner oder löschen Sie sie, denn Sie werden sie sonst noch x-mal öffnen und damit Zeit verlieren.
4. Kennzeichnen Sie auch wichtige Mails und Mails, bei denen eine Antwort unbedingt nötig ist, mit einer hohen Dringlichkeit, und »wünschen« Sie sich von Ihren wichtigen Mailpartnern das Gleiche. Das ist zwar noch nicht sehr üblich, erleichtert die Kommunikation aber ungemein. So sehen Sie auch schon in der Inbox, welche Mails Sie zuerst zuordnen, darüber hinaus hilft es später bei der Suche nach einer Mail.
5. Kennzeichnen Sie Informationsmails und Mails, die nicht unbedingt beantwortet werden müssen, mit einer niedrigen Dringlichkeit. Solche Mails zu beantworten ist Luxus, meist sollten sie nur in eine Ablage wandern.
6. Ändern Sie den Betreff, auch wenn ein Mailwechsel über Tage läuft – geht es um ein anderes Thema, so schreiben Sie das auch in den Betreff.
7. Überlegen Sie zwei Mal, welche Mails als CC und welche als BCC versendet werden sollen. In Arbeitsgruppen ist es immer sinnvoll, ein CC zu verwenden, denn so wissen alle, wer informiert wurde. Einen Witz an 20 Leute zu versenden ist allerdings sicher ein BCC wert.
8. Eine Mail gehört so schnell wie möglich beantwortet und nicht erst nach 2 Tagen. Wenn Sie keine Zeit haben, die gewünschten Informationen zu liefern, oder eine Beantwortung auf Grund von höheren Prioritäten verschieben müssen, dann lassen Sie das Ihren Gegenüber so schnell wie möglich wissen – nicht nur Sie müssen sich Ihre Zeit einteilen: »Ich habe Ihre Mail dankend erhalten und werde sie bis ... beantworten.«

Die erste Abnahme

Die erste Abnahme sollte schon erfolgen, bevor alles läuft. Sie laufen sonst Gefahr, viel Zeit in Details zu stecken, obwohl Sie noch recht falsch liegen.

Stimmen die Farben, und sind die Animationen – grob – so, wie sich das alle gewünscht haben?

Es ist sehr hilfreich, noch nicht die endgültigen Inhalte in die Seite einzubauen. Platzhalter erfüllen meist den gleichen Zweck, und man sieht schon, wie viel Platz für Inhalte vorhanden ist. Nach der Abnahme kann an den Inhalten gefeilt werden, ohne dass die Entwicklung Zeit verschwendet.

Das Gleiche gilt für den Sound. Der verantwortliche Musiker kann sich viel besser in eine Stimmung hineindenken, wenn er schon weiß, wie die Seiten grob aussehen werden und welche Stimmung transportiert werden soll. Die endgültigen Animationen usw. werden dann meistens schon auf den Sound synchronisiert.

Eckpfeiler sind also Design, Animation, danach kommen Inhalt und Sound und erst dann das Feintuning der Seite. Lassen Sie sich die einzelnen Schritte immer abnehmen, je später eine Änderung vom Kunden kommt, umso weiter wirft Sie diese Änderung im Zeitplan zurück.

Alle Kunden sollten anwesend sein

Sorgen Sie dafür, dass alle Entscheidungsträger bei der ersten Abnahme anwesend sind. Es ist zwar auf Kundenseite meist nur eine Person verantwortlich; jetzt sollten aber noch wirklich alle ihre Meinung äußern, das hilft Ihrem Kunden und damit auch Ihrem Team. Bei großen Unternehmen ist es gut, bis zum Vorstand zu gehen, wenn Sie verhindern wollen, dass in der Endphase alles auf den Kopf gestellt wird.

Die Tests

Testen, testen, testen! Bevor Sie mit der Synchronisierung des Sounds beginnen und während Sie noch an den Texten feilen, ist es sinnvoll, schon einige Tests durchzuführen.

- Wie kommt die Zielgruppe mit dem Interface klar?
- Gibt es Logikfehler in der Menüführung?
- Gibt es auf den spezifizierten Systemen (Pflichtenheft!!!) Probleme?
- Überfordern die Animationen die CPU?
- Kommt man mit der Bandbreite klar?

Diese Tests werden allgemein Usability-Tests genannt. Es gibt Firmen, die auf solche Tests spezialisiert sind. Ob Sie nun Freunde innerhalb der Zielgruppe bitten, Ihnen Feedback zu geben oder ein Expertenteam beauftragen, ist dabei weniger wichtig als der Punkt, dass Sie zuhören und lernen. Gerade über harte Kritik sollten Sie sich freuen.

Persönlich rate ich dazu, Experten zu beauftragen, denn das erforderliche Knowhow zu solchen Tests sollte man nicht unterschätzen.

Testen Sie eine Seite immer vor dem Enddesign und bevor Sie an Details feilen.

Fertig, und was jetzt?

Getestet, gefeilt, abgenommen und wieder getestet, gefeilt ... irgendwann ist es soweit: die Endabnahme.

Gute Erfahrungen habe ich damit gemacht, eine kleine Feier zu organisieren, die Daten in einer »feierlichen« Mappe zu übergeben: CD-ROM, Projektdokumentation, Bilder, Rechnung und als kleines Geschenk eventuell noch einen Werbeentwurf für die klassische Presse, denn meistens wird der Kunde seine Seite auch in den klassischen Medien bewerben wollen, und es liegt auch in Ihrem Interesse, dass dies in einer geeigneten Form geschieht.

Alle haben gute Arbeit geleistet, Probleme wurden überwunden, und das Netz ist einmal wieder um eine herausragende Seite reicher.

Optimierung

Flash ist schnell, klein und gut

Flash ist nicht gleich Flash. Gutes Flash unterscheidet sich von schlechtem Flash meist in der Erfahrung des Programmierers. Mit Flash die ersten Animationen zu erstellen ist nicht schwer, sie aber schnell, sicher, klein und sauber zu machen, erfordert mehr.

Tricks für kleine Dateigrößen

▸ Carlo Blatz

Das A und O für kleine Dateien ist natürlich der sinnvolle Einsatz von Symbolen. Jedes Objekt, das mehr als ein Mal gezeigt wird – und das ist bei jedem Tweening der Fall –, sollte ein Symbol sein. Umgekehrt sollten Objekte, die lediglich ein Mal verwendet werden, keine Symbole sein.

Das Verhalten Grafik sollte man im Übrigen ganz vermeiden. Seit Version 2 gibt es einen Bug in Flash, so dass Grafiken unter Umständen mehrmals übertragen werden; jedes Mal, wenn man die Instanz davon verwendet. Und das ist noch nicht genug, außerdem wächst die Dateigröße sogar mit der Framezahl, die eine solche Grafik auf der Zeitleiste bekommt. Bemerkt haben wir diesen Bug bei einem Projekt, in dem der Kunde im Nachhinein sein Timing geändert haben wollte. Wir haben ein Intro auf Kundenwunsch lediglich verlangsamt, also mehr Frames hinzugefügt, und die Dateigröße stieg rapide – ohne dass eine neue Animation oder ein neues Symbol hinzugefügt wurde. Beheben lässt sich dieser Bug ganz einfach: Man verwendet Filmsequenzen statt Grafiken. Was ist eine Filmsequenz aus nur einem Frame? Eine Grafik! Es gibt dadurch keinen Nachteil, und auch wenn der Bug nicht bei jedem Symbol auftritt, sollte man Grafiken doch von vornherein vermeiden. Macromedia weiß seit dem Flash 5-Beta-Test um diesen Flash-Bug; nicht umsonst steht das Verhalten »Filmsequenz« seit Flash 5 an erster Stelle.

Bei Trickfilmen o. Ä. in Flash sind Grafiken natürlich sinnvoll, weil man bereits im Bearbeitungsmodus die Animation im Symbol sehen kann. Man sollte aber auch dann im Final einmal testen, was passiert, wenn man alle Grafiken im Verhalten auf Filmsequenz stellt. Wenn die Anima-

tion im Symbol auch stoppen soll, sobald die Zeitleiste gestoppt wird, kommt man allerdings nicht an Grafiken vorbei.

Ein häufiger Fehler ist die Verwendung der Option Bewegungstween erstellen. Wer hier den Fehler macht, nicht vorher dafür zu sorgen, dass im Anfangs- und Endbild ein Symbol ist, dem setzt Flash eigenständig ins Anfangsbild ein Symbol. Nicht nur, dass dieses »Tween x« (x = laufende Nummer) heißt, es ist zudem auch eine Grafik, was schlecht ist, wie wir wissen. Aber noch schlimmer: Im Endbild hat Flash es beim ursprünglichen Objekt belassen. Das Objekt wird so schon doppelt übertragen.

Dann gibt es natürlich noch diverse Feinheiten. So sollte man nicht automatisch eine Outline um jeden Vektor ziehen, wenn man sie eventuell gar nicht sieht – das sind bei einem Viereck zwar nur wenige Bytes, aber man sollte von vornherein gründlich sein. Warum muss es z. B. im Aktivzustand eines Buttons eine Outline geben? Man sieht sie doch nicht. Wer bezüglich solcher Kleinigkeiten etwas Feingefühl entwickelt, kann so manches KB sparen.

Richtig viele Kilobytes kann man natürlich beim korrekten Import von Pixelbildern sparen (siehe »Tipps zum Import von Pixelbildern« ab Seite 192).

ShapeTweens optimieren

Frei nach dem Motto »Flash ist auch nur ein Mensch« hat das Programm auch seine Grenzen. Die Grenzen bei ShapeTweens sind allerdings recht schnell erreicht. Dennoch gibt es einige Tricks, seine Shapes so zu optimieren, dass sie wie gewünscht aussehen.

Die wichtigste Regel ist die Reduktion von Vektoren auf ein Minimum. Man sollte sogar – wenn es geht – eine zu verformende Grafik in mehrere Teile trennen und diese auf verschiedene Ebenen setzen. So könnte ein zu tweenendes Gesicht in Mund, Nase, Auge, Auge und Hintergrund aufgeteilt werden. Die Chance, dass Flash die Zusammenhänge von vier Vektoren logisch nachvollziehen kann, ist größer, als wenn es um 25 Vektoren geht. Man macht es Flash so etwas einfacher.

Eine andere Möglichkeit, die Animation zu beeinflussen, sind weitere Schlüsselbilder, in denen man die Animation »korrigiert«. Wenn man Formmarken verwendet, muss man beachten, dass auch immer ein neues Anfangsbild der Animation benötigt wird. Ein Korrekturschritt erfordert dann also zwei Schlüsselbilder.

Formmarken können die Animation am besten beeinflussen. Man kann maximal 25 davon setzen und sollte sie im Uhrzeigersinn platzieren. Am effektivsten sind Formmarken an Schnittpunkten und Ecken. Das Prinzip kennt man aus dem »Morphing«.

Flash rechnet auch nur mit Mathematik, und damit gibt es ein paar Prinzipien, die man einhalten sollte, um schöne Animationen zu erstellen. Als beliebtes Beispiel verwende ich in Schulungen ein Formtweening von I nach O. In jedem Formtweening sollten die Vektoren im Anfangs- und im Endbild vergleichbar sein – I und O sind das nicht. Das O hat ein »Loch« im Vektor. Um die Animation überhaupt rechnen zu können, erstellt Flash so im ersten Bild des Tweenings ein riesiges Loch, um dieses dann zum O zu tweenen. Mit einer kleinen Hilfe kann Flash aber eine flüssig laufende Animation erstellen: Dafür erstellt man das erste Bild der Animation von Hand und radiert ein kleines Loch ins I. Nun sind Anfangs- und Endvektor vergleichbar. Den Trick sieht der User nicht.

MotionTweens optimieren

Optimieren heißt beim MotionTween vor allem beschleunigen. Ein optimales MotionTween ruckelt nicht. Bei manchen Animationen lässt sich das zwar nicht verhindern, aber es gibt ein paar Kniffe, die man befolgen kann.

Alphaeffekte sind für den Prozessor und vor allem für die Grafikkarte sehr arbeitsintensiv. Man sollte sie nur im Notfall verwenden. Wir sprechen dabei nicht von kleinen Bannern, sondern von großen Animationen. Fließtexte sind hierfür ein schönes Beispiel, denn dort finden sich viele Vektoren auf großer Fläche. Wenn ein einfarbiger Hintergrund vorliegt, ist die Lösung naheliegend. Man färbt einfach das Symbol in die entsprechende Hintergrundfarbe, und somit ist es unsichtbar. Bei Schwarz und Weiß kann man noch einfacher die Helligkeit einsetzen. Der Effekt ist phänomenal. Auch bei Pixelbildern macht dieser Trick enorm viel aus. Die Frameraten verdoppeln sich bei diesen kritischen Animationen zum Teil.

Aber auch bei verschiedenfarbigen Hintergründen lässt sich oft ein Alphaeffekt vermeiden. Man nimmt einfach einen Wert, der dem Durchschnitt der Hintergrundfarbe nahe kommt. Solange die Animation nur wenige Frames lang ist, bemerkt der User den Trick nicht.

Bei einem Projekt wie www.powerflasher.de/elsa/xmas haben wir sogar festgestellt, dass Bilder mit einem Farbeffekt von Flash deutlich langsamer zu animieren sind als Bilder ohne Farbeffekt. Und weiterhin, dass Vektoren

◀ **Abbildung 1**
Bild 1 von 5 bei einer Animation mit Vermeidung von Alphaeffekt

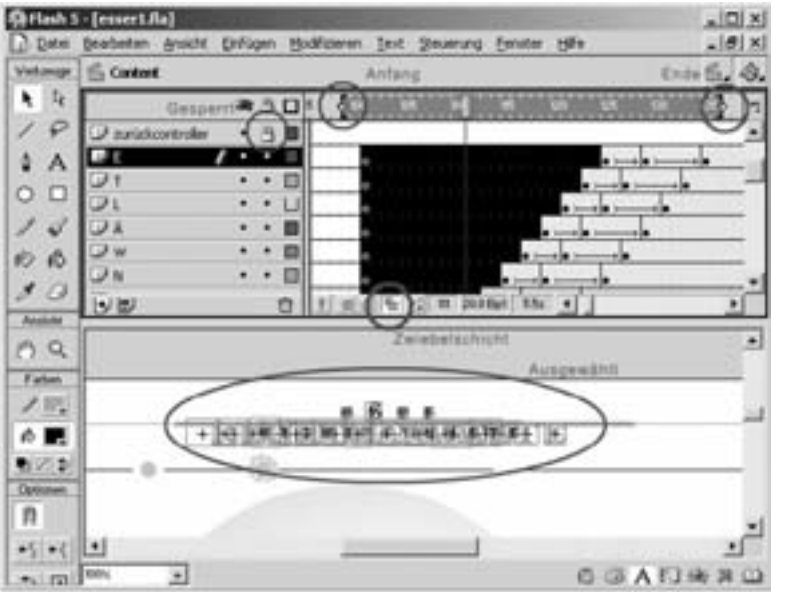
mehr Performance schlucken als Pixelbilder. Natürlich – die Engine von Flash muss ja die Vektoren erst einmal errechnen, darstellen und das gegebenenfalls 30 Mal in der Sekunde. So bauen wir alle großflächigen Animationen möglichst aus Pixelbildern. Der Hintergrund dieses Spiels – die Häuserzeile – ist ein Pixelbild. Wenn man nun den MovieClip, in dem das Pixelbild liegt, einfärbt – nur 10 % Blau hinzugibt oder es um 20 % aufhellt –, nimmt die Performance spürbar ab. Man sollte solche Änderungen also vorher im Grafikprogramm machen.

OnionSkinMarker – der Zwiebelschaleneffekt

Er gehört zwar zu den Standardfunktionen, doch ich treffe immer wieder Flasher, die dieses kraftvolle Tool nicht nutzen. Man verwendet es, um eine Animation über mehrere Frames hinweg zu beeinflussen. So kann man z. B. über 50 Frames alle Objekte verschieben, skalieren, rotieren etc.

Die Anwendung ist denkbar einfach. Man aktiviert das Tool und erhält in der Zeitleiste zwei Marker: den Anfangs- und den Endpunkt. Mit diesen beiden Markern definiert man den Bereich, in dem man arbeiten will,

Abbildung 2 ▶
Ein Texteffekt ist über mehrere Ebenen mit dem Zwiebelschalen-Werkzeug verschiebbar.



sperrt alle Ebenen, die nicht von der Transformierung betroffen sein sollen, und markiert daraufhin alle Objekte (z. B. mit STRG+A – Alles auswählen). Nun kann man die Objekte beliebig transformieren. Achtung, bei vielen Objekten kann das sehr lange dauern, der Prozessor wird stark gefordert. Danach deaktiviert man das Tool und hat eine gleichmäßig transformierte Animation über mehrere Frames. Ohne dieses Tool wäre eine exakte nachträgliche Änderung von Position, Größe o. Ä. kaum oder nur sehr umständlich möglich.

Shared Libraries

Die Shared Libraries sind eine der wohl innovativsten Neuerung in Flash 5. Eine Bibliothek kann mit ihrer Hilfe von verschiedenen Filmen genutzt werden. Das spart Downloadzeit.

Die Anwendungsmöglichkeiten sind vielfältig und werden sich in den nächsten Monaten wohl erst in ihrer ganzen Fülle zeigen. Ich zeige Ihnen einmal ein einfaches, abstraktes Beispiel, das die Funktionsweise demonstriert.

Man nehme ein leeres Dokument und erstelle ein Symbol. Wenn man nun dieses Symbol in der Bibliothek markiert und rechts oben auf die Bibliotheksoptionen klickt, findet man ein Feld Verknüpfung.

In diesen drei Optionen wählen wir die zweite Option Dieses Symbol exportieren. Oben ist nun das Feld `Bezeichner` editierbar. Es ist unbedingt wichtig, dass hier eine aussagekräftige Bezeichnung gewählt wird – sie identifiziert das Symbol später. Achtung: Keine Leerzeichen verwenden.

Unser TestFla können wir nun speichern (sharedlib.fla), als SWF exportieren und beides schließen. Nun öffnet man ein neues FLA und speichert es im gleichen Ordner (animation.fla). Im Menü Datei gibt es den Punkt Gemeinsame Bibliothek öffnen und genau das machen wir nun mit der zuvor abgespeicherten Bibliothek (sharedlib.fla).

Wir sehen nun das erstellte Symbol in der gemeinsamen Bibliothek und verschieben es in die neue Bibliothek. Wenn Sie hier erneut auf Verknüpfungen klicken, können Sie kontrollieren, ob auch der richtige Pfad zur sharedlib.swf angegeben ist. Wenn das nicht der Fall ist, muss man es per Hand nachholen. Einfach den relativen Pfad zur sharedlib.swf angeben und natürlich den Bezeichner nicht vergessen.

Wenn man nun mit diesem importierten Symbol arbeitet, wird die eigentliche Datei nicht viel größer. Testen kann man den Zusammenhang auch, indem man das Symbol in der Shared Library verändert. Das verknüpfte Symbol im Animationsfilm verändert sich auch.

Was man damit alles machen kann, kann man sich lebhaft vorstellen. Ein Logo in fünf Filmen, evtl. sogar über mehrere Frames verteilt, braucht nur noch einmal übertragen zu werden. Man könnte sogar Templates für ganze Seiten bauen. Wenn sich z. B. das Site-Design häufig ändert, braucht man mithilfe der Shared Libraries nur die Bibliothek zu verändern und verändert gleichzeitig die ganze Site.

ActionScript

42 Basics

42 Kritische Gedanken
43 Unterschiede von Flash 4 und 5

46 Grundlagen von ActionScript

46 IF-Anweisung
48 Schleifen
51 Objekte
57 STRING-Manipulationen

62 Objektorientierte Programmierung in Flash

67 Der Array-Zugriffsoperator []
69 Der With-Operator
70 Selbstdefinierte Objekte erstellen
73 Die Eigenschaft Prototype
75 Eigendynamik mit Zufallswerten erzeugen

80 Bézierkurven in Flash

80 Aufbau des Films

86 Beispiele für den Einsatz von ActionScript

86 Yugop-Menü in Flash 5

92 Starfield-Simulation in Flash 5

94 Flash 5 Draw

98 Smart-Filmsequenzen

99 netTrekMenu

100 Wie funktioniert meine Smart-Filmsequenz?

Basics

Die interne Programmiersprache von Flash

Flash 5 ohne ActionScript, das ist wie ein Flugzeug ohne Elektronik. Auf was es ankommt und was ActionScript für Flash 5 überhaupt ist, werden wir hier zusammenfassen.

Kritische Gedanken

▸ Gerald Marischka

ActionScript ist kein Allheilmittel, es ist eine Form der Programmierung.

Es ist nicht immer sinnvoll, möglichst viel Technik in eine Seite einzubauen, Ziel muss es sein, mit dem geringsten Aufwand eine elegante, funktionierende und platzsparende Lösung zu finden.

Es kommt sicherlich oft vor, dass man ActionScript verwendet, gerade weil ActionScript diese elegante und platzsparende Form bieten kann. Aber ist es auch immer sinnvoll?

Wenn man bedenkt, welche Schockwelle die Bugs der ersten Flash 5-Plug-ins unter den Entwicklern ausgelöst haben, so kann man zu einer bedingungslosen ActionScript-Entwicklung nicht wirklich raten. Nun kommen die Probleme sicherlich nicht nur vom ActionScript-Einsatz, jedoch ist es ein Fakt, dass alte Flash 3- und 4-Seiten, mit Tell Target und nicht mit ActionScript entwickelt, problemlos laufen. Es ist unter diesem Aspekt durchaus ratsam, den Weg über die klassischen Methoden (Tell Target, Tweens usw.) zu gehen, bevor man sich der Gefahr aussetzt, die Seiten nach dem Erscheinen einer neuen Flash-Version teilweise neu programmieren zu müssen.

Es bleibt zu wünschen, dass die doch umfangreichen Änderungen von Flash 4 zu 5 uns trotzdem eine »unproblematische« Abwärtskompatibilität erlauben, denn Kompatibilität war bisher einer der großen Pluspunkte von Flash.

So wie sich Flash verändert hat, so müssen sich auch die Experten anpassen. Zwischen den verschiedenen Versionen von Flash ist eine deutliche Verlagerung zu beobachten. Der Trend geht weg vom grafikorientierten Programm und hin zum klassischen Programmieren.

Es ist nicht mein Ziel, auf Vor- oder Nachteile dieses Trends einzugehen. Fakt ist aber, dass Sie als Profi in Zukunft nicht ohne ActionScript auskommen werden, denn das neue Flash 5 ActionScript ermöglicht neue, schnellere Wege und Möglichkeiten, an denen Sie in Zukunft nicht vorbeikommen werden.

Es freut mich ganz besonders, dass der Berater meines Teams in Sachen »Freak-Programmierung« und mein persönlicher Freund, Christoph Aigner, Ihnen anhand dieses Kapitels sein Wissen näher bringen wird.

Unterschiede von Flash 4 und 5

◀
Christoph Aigner & Gerald Marischka
www.alaris.at

Flash 5 definiert sich durch eine komplett überarbeitete Skriptsprache. Die Sprache lehnt sich stark an bestehende Sprachen an (JavaScript, PHP).

Die Vorteile liegen klar auf der Hand: Wenn man eine Sprache und einen Programmieransatz gelernt hat, dann fällt der Umstieg auf ActionScript nicht weiter schwer. Die Elemente sind in gewohnter Weise anwendbar, und die Logik bleibt gegenüber verwandten Sprachen identisch. Für Programmierer ist also ein leichter Einstieg in Flash 5 möglich.

Designer und Anwender von Flash 4 müssen allerdings umdenken. Folgende Charakteristika wurden in Flash 5 umgesetzt:

- Ansprechen der Elemente über Punktnotation:
  `_root.movie1._eigenschaft1`
- Systematischer Aufbau der Hierarchiestufe von Objekten (wenn ein Movie an ein anderes per `attachMovieClip` angehängt wird, kann dies direkt über die Erweiterung eines bestehenden Knotens angesprochen werden):
  `root.movie1.movie2._eigenschaft2`
- Neueinführung von Objekten (Objekte z. B. vom Typ `Date` können instanziiert und manipuliert werden):
  ```
  MyDate = new Date();
  MyDate.SetMinutes = 50;
  ```
- Die Verwendung von Funktionen ermöglicht eine effiziente und logische Programmierung z. B. dadurch, dass wiederholte Abläufe strukturiert zusammengefasst werden.
- Klassen können durch die Einbindung von MovieClips simuliert werden, da MovieClips Funktionen, Werte und Eigenschaften enthalten können.

- XML-Socket/XML-Strukturen ermöglichen beständige Verbindungen zu Servern und eine logische Bereitstellung von Verknüpfungen und Kommunikationsschnittstellen.
- Keine Einschränkung der ausgeführten Aktionen mehr (es gibt keine Limitierung mehr, wie das noch bei Flash 4 pro MovieClip der Fall war).
- Manipulation aller Movie-Eigenschaften während der Laufzeit (ermöglicht starke Erhöhung der Interaktion mit dem Benutzer).

Grundlagen von ActionScript

Ein bisschen Theorie muss sein

Im Folgenden werde ich für Sie die Grundlagen und Besonderheiten von ActionScript wiederholen. Für einen Nichtprogrammierer mag der ganze ActionScript-Teil anfangs etwas trocken wirken, doch wenn man das Grundgerüst und den Kern einer Programmiersprache begriffen hat, dann lässt sich mit Geschick und Fantasie jede Idee verwirklichen.

▸ Christoph Aigner
www.alaris.at

Ich komme aus der reinen Programmierwelt (angefangen von C/C++ über Java bis hin zu den Internet-Skriptsprachen PHP, CGI, ASP) und bin fasziniert von den Möglichkeiten in Flash 5, wo Designer und Programmierer mit ein und demselben Tool arbeiten.

Dieser Teil setzt auf dem Benutzerhandbuch von Flash 5 auf und stellt somit kein Repetitorium aus gelesenem Material dar, sondern führt effizient in die Programmierwelt ein. Beginnend mit elementaren Funktionen und Interaktionen werden die Chancen und Möglichkeiten von ActionScript aufgezeigt.

Um Variablen und Objekte manipulieren und steuern zu können, kommen oft Anweisungen und Schleifen zum Einsatz. Folgende Abfragen und Regelwerke stehen zur Verfügung:

- IF-Anweisungen
- FOR-Schleifen
- FOR-IN-Schleifen
- WHILE-Schleifen
- DO-WHILE-Schleifen

IF-Anweisung

Die IF-Anweisung ist der am häufigsten gebrauchte Befehl in der ActionScript-Programmierung. Er dient dazu, den Status (Wert) einer Variablen zum aktuellen Zeitpunkt abzufragen. Die allgemeine Schreibweise lautet:

```
If (AUSDRUCK1 OPERATOR AUSDRUCK2)
```

Dazu können folgende Operatoren verwendet werden:

- == Der 1. Ausdruck ist identisch mit dem 2. Ausdruck.
- != Der 1. Ausdruck ist verschieden (ungleich) vom 2. Ausdruck.
- < Der 1. Ausdruck ist kleiner als der 2. Ausdruck.
- > Der 1. Ausdruck ist größer als der 2. Ausdruck.
- <= Der 1. Ausdruck ist kleiner oder gleich dem 2. Ausdruck.
- >= Der 1. Ausdruck ist größer oder gleich dem 2. Ausdruck.

Wenn die Aussage in der IF-Anweisung wahr ist, wird die nachstehende Aktion ausgeführt. Falls es notwendig ist, mehrere Aktionen bezogen auf eine IF-Anweisung auszuführen, muss der nachstehende Befehlsblock in geschwungene Klammern gesetzt werden.

```
if (VARIABLE1 > EVALUATIONSWERT) {
// wird nur ausgeführt, falls IF-Anweisung erfüllt ist.
     BEFEHL1;
     BEFEHL2;
}
```

Zudem besteht die Möglichkeit, einen Anweisungsblock anzuhängen, falls der Inhalt der IF-Anweisung nicht erfüllt ist. Dies geschieht in folgender Form:

```
else {
// wird nur ausgeführt, falls IF-Anweisung nicht erfüllt ist.
    BEFEHL3;
    BEFEHL4;
}
```

Verkettung von IF-Anweisungen

Bei Notwendigkeit der Evaluierung (Überprüfung) mehrerer Aussagen innerhalb einer IF-Anweisung können diese Aussagen durch eine logische Verknüpfung aneinander gereiht werden. Die allgemeine Schreibweise lautet:

```
If (BEFEHLSBLOCK1 VERKNÜPFUNG1 BEFEHLSBLOCK2 .. BEFEHLSBLOCKx VER-
KNÜPFUNGy BEFEHLSBLOCKz)
```

Folgende Verknüpfungsmöglichkeiten stehen dem Anwender zur Verfügung:

- && (auch »and« möglich), der erste und der zweite Befehlsblock müssen erfüllt sein
- || (auch »or« möglich), der erste oder der zweite Befehlsblock müssen erfüllt sein

Man kann Befehlsblöcke auch durch normale Klammernsetzung gruppieren und verschachteln.

Schleifen

Schleifen dienen der wiederholten Ausführung von Programmteilen (Anweisungsblöcken). Jeder Schleifentypus weist einen Schleifenkopf auf, der durch eine Bedingung gekennzeichnet ist. Diese Bedingung legt die Anzahl der Aufrufe des Anweisungsblocks fest. Wenn die Bedingung einer Schleife nicht erfüllt ist, wird die Schleife ignoriert und die Programmausführung nach dem Funktionsblock der Schleife fortgesetzt. Der Funktionsblock wird, wie bei der IF-Anweisung, mit geschwungenen Klammern gekennzeichnet.

Falls ein vorzeitiges Abbrechen der Iteration durch den Funktionsblock einer Schleife erwünscht ist, so kann dies durch einen Aufruf der Funktion `break();` erzielt werden.

FOR-Schleife

FOR-Schleifen dienen der in der Anzahl der Durchläufe festgelegten, wiederholten Ausführung eines Anweisungsblocks in ActionScript. Der Aufbau einer FOR-Schleife hat folgendes allgemeines Aussehen:

```
For (VARIABLENSTARTWERT; BEDINGUNG; VARIABLENÄNDERUNG) {
// auszuführender Anweisungsblock
}
```

Ein konkretes Beispiel wäre:

```
For (i = 0; i < 10; i++) {
trace ("Aktueller Wert: " + i);
}
```

Das erste Argument der FOR-Schleife setzt die Zählervariable (Variable, welche die Anzahl der Durchläufe der Schleife steuert) auf einen Startwert. Danach kommt die Bedingung, die erfüllt sein muss, damit das Programm den nachstehenden Anweisungsblock ausführt. Dieser Bedingungsblock kann wie bei der IF-Anweisung strukturiert werden. Als drittes Argument übergibt man der Schleife die Wertänderung pro Durchlauf. In dem oben angeführten Fall wird der Zähler (`i`) pro Durchlauf um 1 erhöht. Statt `i++` kann auch `i = i + 1` geschrieben werden.

FOR-IN-Schleife

Dieser Schleifentypus stellt eine spezielle Form der FOR-Schleife dar. Die allgemeine Struktur sieht folgendermaßen aus:

```
For (EIGENSCHAFT in OBJEKT) {
// auszuführender Anweisungsblock
}
```

Der Anweisungsblock wird für jeden der Positionen (Eigenschaften) innerhalb des Objektes ausgeführt. Das Objekt kann ein Array oder ein Movie sein. Zur Veranschaulichung folgendes Beispiel:

```
myObject = { Hoehe:'5 cm', Breite:'20 cm', Farbe:'Grün' };
for (attribut in myObject) {
trace ("myObject." + attribut + " = " + myObject[attribut]);
}
```

Dies gibt Folgendes aus:

```
myObject.Hoehe = 5 cm
myObject.Breite = 20 cm
myObject.Farbe = Grün
```

Wie das Beispiel zeigt, wird das Objekt `myObjekt` zur Gänze durchlaufen und jedes einzelne Attribut des Objektes inklusive dem Wert des Attributes im Output-Fenster ausgegeben. Auf diese Weise kann man rasch jede einzelne Eigenschaft eines Objektes um dasselbe Attribut verändern.

WHILE-Schleife

Die WHILE-Schleife stellt die flexibelste aller Schleifen dar. Der Anweisungsblock dieser Schleife wird so lange wiederholt, wie die Bedingung(en) der Schleife erfüllt sind.

Die WHILE-Schleife hat folgende Struktur:

```
While (BEDINGUNG) {
// auszuführender Anweisungsblock
}
```

Mehrere Bedingungen können wiederum durch || oder && verbunden werden.

Denken Sie ans Abbrechen!
Dieser Schleifentypus führt des Öfteren zu Endlosschleifen, da mancher Programmierer schon vergessen hat, die Schleife durch eine Nichterfüllung der Bedingung abzubrechen.

DO-WHILE-Schleife

Die DO-WHILE-Schleife ist bis auf eine Ausnahme identisch zur gewöhnlichen WHILE-Schleife. Der einzige Unterschied besteht darin, dass bei DO-WHILE die Bedingung am Ende des Anweisungsblocks steht und nicht zu Beginn des auszuführenden Blocks. Sie ist folgendermaßen gekennzeichnet:

```
Do {
// auszuführender Anweisungsblock
} while (BEDINGUNG);
```

Ansonsten gelten dieselben Eigenschaften, die auch bei der WHILE-Schleife Gültigkeit finden.

Objekte

Selection-Objekt

Das Selection-Objekt dient der Setzung des Eingabe-Auswahl-Fokus. Über die Funktion `Selection.getFocus` erhalten Sie den absoluten Pfad zum augenblicklich fokussierten Element.

Es empfiehlt sich, bei Flash 5 die Tabulatorreihenfolge selbst festzulegen, da die automatische Reihenfolge bei größeren Eingabemasken zur Verwirrung führt. Dies geschieht folgendermaßen:

Zu Beginn wird folgendes Array anhand der zur Verfügung stehenden Objekte definiert (siehe Arrays):

```
Tabs = new Array();
Tabs[Tabs.length] = _root.window.text1;
Tabs[Tabs.length] = _root.window.text2;
Tabs[Tabs.length] = _root.window.text3;
Tabs[Tabs.length] = _root.window.button;
```

In der Frameleiste wird die Funktion `TABSET` definiert:

```
function TABSET() {
   if (this.TABPOSITION >= Tabs.length)
        this.TABPOSITION = 0;
Selection.setFocus(Tabs[this.TABPOSITION]);
}
```

Bei dem zugehörigen MovieClip wird folgender Code implementiert:

```
onClipEvent(enterFrame) {
if(Key.isDown(Key.TAB) || Key.isDown(Key.ENTER)) {
       this.TABPOSITION++;
       this.TABSET();
   }
}
```

Dieses Beispiel übernimmt die Tabulatorensetzung (Reihenfolge) für eine Eingabemaske.

Sobald TAB oder ENTER gedrückt wird, wird die aktuelle Tabulatorposition um eins erhöht und die Funktion TABSET aufgerufen. In der Funktion TABSET wird kontrolliert, ob die aktuelle Tabulatorposition außerhalb des Arrays liegt. Falls dies der Fall ist, beginnt der Tabulator wieder beim ersten Element den Fokus zu setzen.

Date-Objekt

Über das Date-Objekt ist es möglich, Daten- und Zeitstrukturen entweder relativ zur Greenwich Mean Time (= Universal Coordinated Time) oder zur Zeit des jeweiligen lokalen Systems abzubilden. Ersteres ist insbesondere für international gesteuerte Prozesse interessant, da durch die Universal Coordinated Time Zeitzonen der einzelnen Länder berücksichtigt werden können. Die kleinste Zeiteinheit des Date-Objekts ist eine Millisekunde, wodurch sehr kleine Zeitintervalle gesteuert werden können.

Ein Beispiel für die Anwendung des Date-Objekts in Bezug auf die Universal Coordinated Time wäre eine weltweite Firmenpräsentation einer Neuheit, die in den einzelnen Ländern zur selben Zeit starten soll.

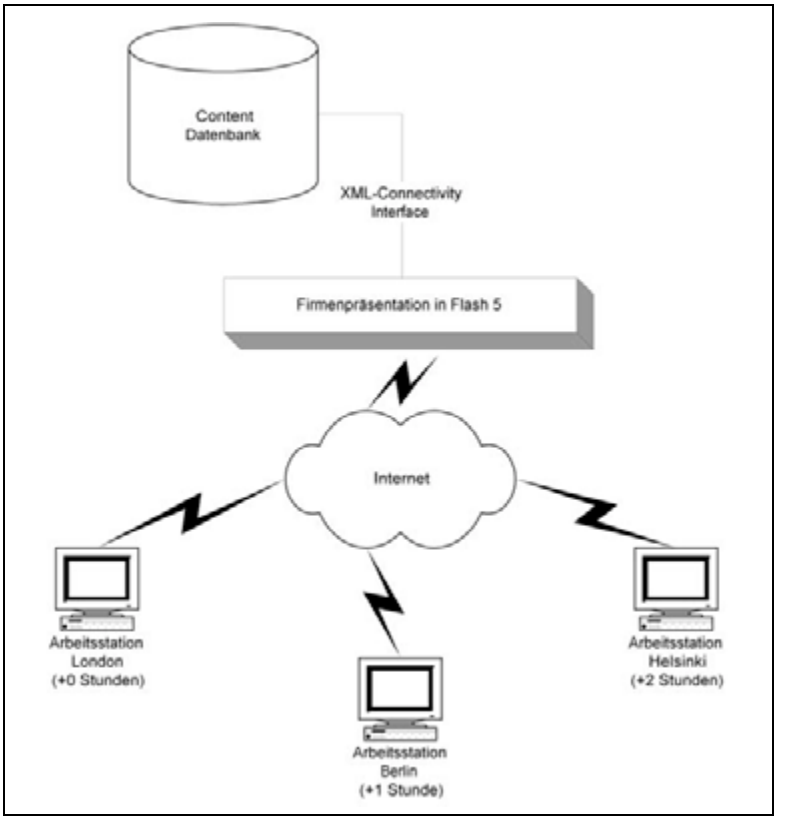
Um obige Firmenpräsentation zu realisieren, werden folgende Funktionen des Date-Objekts benötigt: **Constructor, getTime, setFullYear, setMonth, setDate, setHours, setMinutes.**

Setzen des Startdatums einer Weihnachtspräsentation für Firma X am 24.12.2001 um 18.30 lokale Zeit:

```
PresentationStart = new Date();
PresentationStart.setFullYear(2001);
PresentationStart.setMonth(12);
PresentationStart.setDate(24);
PresentationStart.setHours(18);
PresentationStart.setMinutes(30);
PresentationsStartUTC = PresentationStart.getTime();

Now = new Date();
NowUTC = Now.getTime();

if(NowUTC < PresentationsStartUTC) {
  trace("Weihnachten ist noch nicht...");
} // of if
else{
```



◀ **Abbildung 1**
Internationale Firmenpräsentation

```
  trace("Frohe Weihnachten!");
trace("Zeit in London: " + Now.getUTCHours()+":"+Now.getUTC
Minutes()");

midnight = new Date();
midnight.setUTCFullYear(2001, 12, 24);
midnight.setUTCHours(23,59,59,59);
difference = midnight.getTime()-Now.getTime();
mindifference = difference/60000;
trace("In London haben die Menschen noch "+mindifference+"Minuten
Zeit, um ins Bett zu gehen");
}// of else
```

Zu Beginn wird das Date-Objekt mit dem Referenzhandler `PresentationStart` instanziiert. (Es wird ein Objekt erzeugt, das dieselben Eigenschaften und Funktionen aufweist, die im Date-Objekt definiert sind.) Standardgemäß wird das Date-Objekt beim Instanziieren mit der lokalen Systemzeit initialisiert. Da der Starttermin der Präsentation jedoch auf den 24.12. lokale Zeit festgelegt werden soll, verwenden wir die Funktionen `setFullYear, setMonth, setDate, setHour` und `setMinute`, um den Zeitpunkt zu definieren. Man könnte sich auch die Funktionen `setMonth, setHour` und `setMinute` ersparen, verwendete man gleich die optionalen Parameter von `setFullYear` und `setHours`. Weiter unten im Beispiel ist die Verwendung der optionalen Parameter bei den Funktionen `setUTCFullYear` und `setUTCHours` illustriert.

Mit der Variablen `Now` erhalten wir nach Instanziierung des Date-Objekts die lokale Systemzeit, die wir mit der Funktion `getTime` in ein UTC-Format umrechnen. Die Funktion `getTime` gibt uns die UTC-Millisekunden seit 1.1.1970, 0:00, des Date-Objekts zurück.

In der ELSE-Schleife wird gezeigt, dass man mit dem Date-Objekt auch ganz einfach Zeitdifferenzen errechnen kann. Da die Zeit in Millisekunden von der Funktion `getTime` zurückgegeben wird, muss die Differenz durch 60000 (= 60x1000) dividiert werden, um auf Minuten zu kommen. Einen ausführlichen Workshop finden Sie auf Seite 542.

Array-Objekt

Mit dem Array-Objekt ist es möglich, n-dimensionale Raumausprägungen in einer Variablen zu speichern. Auf diese Art können z. B. n-dimensionale Würfel generiert werden, um Daten nach unterschiedlichen Ausprägungsdimensionen zu speichern. Dadurch können (wie im nachfolgenden Beispiel anschaulich dargelegt) Daten in logische Einheiten zusammengefasst und so auf simple Art und Weise manipuliert werden.

Beispielsweise können so in einem Würfel die Umsätze von Firmen nach den unterschiedlichen Regionen in Abhängigkeit von Produkt, Produktgruppe und Zeitraum abgelegt werden, um diese dann über eine grafische Anzeige zu visualisieren.

Die Umsätze der Firma X (Werte in tausend EUR):

Produkte/Monat	Januar	Februar	März	April
Software				
Textverarbeitung	58,2 (D) 40,3 (E)	60,8 (D) 45,5(E)	40,1 (D) 50,8 (E)	80,2 (D) 45,9 (E)
Lohnverrechnung	35,8 (D) 50,9 (E)	50,2 (D) 40,8 (E)	54,5 (D) 38,2 (E)	67,2 (D) 26,5 (E)
Hardware				
Computer	321,3 (D)	324,21 (D)	354,4 (D)	380,6 (D)
Notebooks	140,9 (D) 210,4 (E)	150,5 (D) 234,6 (E)	130,3 (D) 265,3 (E)	145,2(D) 290,4 (E)
Monitore	100,4 (D)	114,8 (D)	120,43 (D)	140,3 (D)

Um obige Umsatztabelle nun in Flash abzubilden und die Daten in Dimensionen effizient bearbeiten zu können, benötigt man ein vierdimensionales Array.

Um die Performance der Arrayoperatoren zu erhöhen, ersetzen wir die einzelnen Strings durch Zahlenwerte. Folgende Definitionen gelten:

0.. Produktkategorie Software, 1..Hardware
0.. Textverarbeitung, 1..Lohnverrechnung, 2.. Computer, 3.. Notebooks, 4.. Monitore
0.. Deutschland,1.. England
1.. Januar, 2.. Februar, 3.. März, 4.. April

Diese weisen wir Variablen zu, um den Beispielcode einfacher lesbar zu machen:

```
cSoftware = 0;
cHardware = 1;
cTextverarbeitung = 0;
cLohnverrechnung = 1;
cComputer = 2;
cNotebook = 3;
cMonitore = 4;
cDeutschland = 0;
cEngland = 1;
cJan = 1;
cFeb = 2;
cMarch = 3;
cApr = 4;
```

Um die Textverarbeitungsumsätze im Array `UmsaetzeX` zu speichern, wird folgende Implementierung benötigt:

```
UmsaetzeX = new Array();
UmsaetzeX[cSoftware] = new Array();
UmsaetzeX[cSoftware][cTextverarbeitung] = new Array();
UmsaetzeX[cSoftware][cTextverarbeitung][cJan] = new Array();
UmsaetzeX[cSoftware][cTextverarbeitung][cJan][cDeutschland] = 58.2;
UmsaetzeX[cSoftware][cTextverarbeitung][cJan][cEngland] = 40.3;
```

Das Array muss, wie im obigen Beispiel gezeigt, für jede neue Dimension initialisiert werden. Liest man die Werte aus einer XML-Datenquelle, so wird man in der Praxis natürlich auf die konstanten Variablen verzichten und ein n-dimensionales Array automatisch mit einer Iteration generieren.

Angenommen wir haben wie im obigen Beispiel auch die Umsätze der Firma Y und Z im Array `UmsaetzeY` und `UmsaetzeZ` gespeichert. Nun wollen wir die Umsätze aller Firmen in einem Array zusammenführen. Dies wird in Flash über die Funktion `concat` bewerkstelligt. Zu beachten ist jedoch, dass alle Arrays die gleiche Anzahl an Dimensionen besitzen.

```
UmsaetzeY = new Array();
UmsaetzeZ = new Array();
..
//Zuweisen der Werte
..
UmsaetzeGlobal = UmsaetzeX.concat(UmsaetzeY, UmsaetzeZ);
```

Über die JOIN-Funktion können alle Elemente eines Arrays in einer Stringvariablen gespeichert werden. Um die einzelnen Elemente im String noch auseinander halten zu können, kann ein Separatorstring übergeben werden. In unserem Beispiel verwenden wir das +-Zeichen als Trennungsstring.

```
MyOutput = UmsaetzeX.join(" + ");
trace(MyOutput);
```

Möchten wir beispielsweise das Umsatzarray der Firma X nach den Textverarbeitungsumsätzen sortieren, um beispielsweise immer das Land mit

dem höchsten Umsatz am Schluss zu haben, so benötigen wir folgende Implementierung:

```
UmsaetzeJan = new Array();

UmsaetzeJan[0] = UmsaetzeX[cSoftware]
[cTextverarbeitung][cJan][cDeutschland];
UmsaetzeJan[1] = UmsaetzeX[cSoftware]
[cTextverarbeitung][cJan][cEngland];

UmsaetzeJan.sort();
trace(UmsaetzeJan.join(" ,"));
```

Zuerst werden die Eckpunkte des Umsatzraumes in ein sortierbares Array kopiert, anschließend über die SORT-Funktion nach Umsatz sortiert und ausgegeben.

STRING-Manipulationen

Um einen String (Textkette) zu manipulieren, werden Methoden des Objektes `String` angewandt. Normalerweise ist es nicht notwendig, ein Objekt des Typus `String` zu erzeugen, um eine Textkette zu untersuchen und manipulieren. Doch besteht auch die Möglichkeit, eine Instanz von einer Textkette zu schaffen, um diese dann für die weitere Bearbeitung zu verwenden. Im Folgenden wird die Funktionsweise der Befehle anhand einiger Anwendungsmöglichkeiten aufgezeigt.

Stringaddition (String.concat)

Mehrere Zeichenketten können mit Hilfe des Zeichens + aneinander gekettet werden.

```
Final = str1 + str2;
```

So ist es möglich, beliebig lange Zeichenketten ohne größeren Aufwand in eine neue einzelne Zeichenkette zu kopieren. Dasselbe Ziel wird durch die Anwendung der Funktion `String.concat` erreicht.

String.charAt(index) String.charCodeAt(index) String.fromCharCode(c1, c2, cN)

Überprüfung, ob ein Zeichen in einem String vorhanden ist, am Beispiel einer E-Mail-Adresse:

```
str = max@mustermann.com
vorhanden = false;
for (i = 0; i < str.length; i++) {
       if (str.charAt(i) == "@")
            vorhanden = true;
}
```

Dieses Beispiel prüft zeichenweise, ob der Text `str` ein @-Zeichen enthält. Sobald ein Zeichen einem @ entspricht, wird die Variable `vorhanden` auf `true` (wahr) gesetzt. Die zeichenweise Überprüfung erfolgt mit Hilfe einer FOR-Schleife. Die aktuelle Position innerhalb der Zeichenkette wird durch `i` ausgedrückt. Die Schleife wird vom ersten Zeichen (der Index eines Strings beginnt mit 0) bis zum letzten Zeichen (`str.length` gibt die Länge der Zeichenkette zurück) untersucht. Durch den Aufruf der Funktion `charAt(i)` wird das Zeichen an der Position `i` ermittelt.

Die Funktion `String.charCodeAt` unterscheidet sich von der Funktion `String.charAt` dadurch, dass der zurückgelieferte Wert nicht das Zeichen selbst, sondern die Repräsentation als Zahl bezogen auf den ASCII-Code darstellt.

Als Gegenstück zu `String.charCodeAt` existiert die Funktion `String.fromCharCode`. Diese Funktion wandelt x-beliebig viele, durch Beistriche getrennte Zeichen im ASCII-Code-Format in einen ganzheitlichen String um.

String.split(delimeter)

Aufteilen einer Zeichenkette anhand eines Trennzeichens in mehrere Teile, die in einem Array gespeichert werden (am Beispiel einer Keyword-Suche):

```
str = "Farben,Blau,Grün,Gelb,Rot,Mischfarben";
keywords = new Array();
keywords = str.split(",");
```

Es wird demonstriert, wie einfach ein String in ein Array aufgeteilt werden kann, wenn ein Trennzeichen zur Verfügung steht. Ein geeignetes Anwendungsgebiet wären Suchmaschinen.

String.substr(start,length) String.substring(from, to)

Entfernen der letzten vier Zeichen eines Strings anhand eines HTML-Dokuments:

```
str = "Hallo Welt<br>";
final = str.substr(0, str.length - 4);
```

Das obige Beispiel schneidet die letzten vier Zeichen eines Strings ab, um den ursprünglich enthaltenen HTML-Code zu entfernen. Durch den Aufruf von `str.substr(0, str.length – 4)` wird die ganze Zeichenkette minus der letzten vier Zeichen (durch `str.length-4` ausgedrückt) zurückgeliefert.

Die Funktion `String.substring` unterscheidet sich von der Funktion `String.substr` dadurch, dass bei `String.substring` als zweiten Parameter die Länge übergeben wird, die der auszuschneidende String haben soll.

String.toLowerCase() String.toUpperCase()

Überprüfen eines Benutzernamens anhand einer nicht case-sensitiven Benutzerliste:

```
list = new Array();
list[0] = "test";
list[1] = "Chrisu";
list[2]  = "Waldi";

eingabe = "CHRISU";
erfolg = false;

for (i = 0; i < list.length; i++)  {
if (eingabe.toLowerCase() == list.toLowerCase())
    erfolg = true;
}
```

Das obige Beispiel vergleicht den String `eingabe` mit der Liste `list`, um festzustellen, ob der User `CHRISU` in der Liste existent ist. Falls der Wert in `eingabe` gefunden wird, erfolgt die Zuweisung `erfolg = true`. Um sicherzustellen, dass es egal ist, ob der Benutzername in Großbuchstaben eingegeben worden ist oder in der gleichen Schreibweise wie in der Liste, werden alle Buchstaben in Kleinbuchstaben umgewandelt, die dann zueinander überprüft werden.

Im Gegensatz dazu konvertiert die Funktion `String.toUppercase` den gesamten Inhalt der Strings in Großbuchstaben.

Objektorientierte Programmierung in Flash

Seit Flash 5 hat sich einiges an ActionScript geändert. In Flash wird nun objektorientiert programmiert. Flash 5 beherrscht aus Kompatibilitätsgründen trotzdem die alte Slash-Syntax und die Funktionen aus Flash 4, sie gelten aber als veraltet. Es ist empfehlenswert, auf die alte Syntax komplett zu verzichten.

▸ Carlo Blatz

Objekte begegnen uns in Flash wie in jeder anderen höheren Sprache an allen nur erdenklichen Stellen. So ist z. B. jeder MovieClip, den wir erstellen, ein Objekt des Typs MovieClip. Und jeder MovieClip, den wir auf der Bühne platzieren, ist eine Instanz des MovieClip-Objekts.

In Flash 5 sprechen wir mit Hilfe der Dot-Syntax alles über Objekte an. Im Folgenden zeige ich Ihnen, wie man Eigenschaften von Objekten setzt und die Methoden aufruft.

Die Flash 4-Bedingung, um eine Instanz auf der aktuellen Zeitleiste von Bild 1 abspielen zu lassen, war folgende:

```
TellTarget ("Instanz")
  GotoAndPlay(1)
End TellTarget
```

Sie lässt sich in Flash 5 durch ein simples

```
Instanz.gotoAndPlay(1);
```

erledigen.

Doch welche Pfade gibt es jetzt?

- Zunächst einmal sind da die **Levels**, mit denen man mehrere per `LoadMovie` geladene Filme übereinander und untereinander schachteln kann. So wird der `MovieClip` mit dem Instanznamen `Instanz` in Level 99 angewiesen, zu Bild 1 zu springen und abzuspielen:
  `_level99.Instanz.gotoAndPlay(1);`

- Ein neues Ziel gibt es jetzt: `_root`. Es entspricht dem »/« aus Flash 4 – die Hauptzeitleiste des aktuellen Levels.
  `_root.Instanz.gotoAndPlay(1);`
- Mit dem Alias `_parent` referenziert man den MovieClip, der den aktuellen MovieClip beinhaltet. In Flash 4 erfolgte dies mit ../

Das geht natürlich nicht nur mit Zeitaktionen wie `Goto, Stop, Play`. In Flash 5 kann man nun so jegliche Eigenschaften setzen, wie es in Flash 4 nur über `SetProperty` ging. So würde z. B. die Instanz auf der Hauptzeitleiste des aktuellen Films auf die X-Position 100 gesetzt:

```
_root.Instanz._x = 100;
```

So könnte man dann die gewünschte Instanz mit einer Variablen verknüpfen:

```
_level99["Instanz" + variable]._y = variable * 100;
```

Wenn die Variable also immer um 1 erhöht wird, würde `Instanz1` bei y = 100, `Instanz2` bei y = 200 etc. positioniert werden.

Praxisbeispiel

Basteln wir einen MovieClip, der die folgende Struktur hat:

- hauptClip
- Der hauptClip besitzt einen hauptUnterClip.
- Dieser hauptUnterClip besitzt drei unterClips: unterClip1, unterClip2, unterClip3.
- In jedem der drei unterClips sind jeweils fünf Variablen definiert. hauptClip besitzt anders ausgedrückt also folgende Struktur:

```
- hauptClip
   hauptUnterClip
      unterClip1
         var1
         ...
         var5
      unterClip2
         var1
         ...
```

```
      var5
   unterClip3
      var1
      ...
      var5
```

Von der Hauptzeitleiste aus soll `unterClip1` starten und die X-Position von `unterClip2` verändern:

Flash 4

```
Begin Tell Target ("hauptClip/hauptUnterClip/unterClip1")
      Go to and Play ("StartMarke")
End Tell Target
Set Property ("hauptClip/hauptUnterClip/unterClip2", X-Position) =
GetProperty ("hauptClip/hauptUnterClip/unterClip2", _x) + 10
```

Flash 5

```
hauptClip.hauptUnterClip.unterClip1.gotoAndPlay ("StartMarke");
hauptClip.hauptUnterClip.unterClip2._x += 10;
```

Von der Hauptzeitleiste aus soll die Variable `var1` in `unterClip1` auf 0 gesetzt werden:

Flash 4

```
Set Variable: " hauptClip/hauptUnterClip/unterClip1/:var1" = 0
```

Flash 5

```
hauptClip.hauptUnterClip.unterClip1.var1 = 0;
```

Von `unterClip1` aus soll `unterClip2` gestartet und `unterClip3` unsichtbar gemacht werden.

Flash 4
Absolut

```
Begin Tell Target ("/hauptClip/hauptUnterClip/unterClip2")
      Go to and Play ("StartMarke")
End Tell Target
Set Property ("/hauptClip/hauptUnterClip/unterClip3",
Sichtbarkeit) = false
```

Relativ

```
Begin Tell Target ("../unterClip2")
      Go to and Play ("StartMarke")
End Tell Target
Set Property ("../unterClip3", Sichtbarkeit) = false
```

Flash 5
Absolut

```
_root.hauptClip.hauptUnterClip.unterClip2.gotoAndPlay
("StartMarke");
_root.hauptClip.hauptUnterClip.unterClip3._visibility = false;
```

Relativ

```
_parent.unterClip2.gotoAndPlay ("StartMarke");
_parent.unterClip3._visibility = false;
```

Von `unterClip1` aus soll die Variable `var1` in `unterClip2` auf 0 gesetzt werden.

Flash 4
Absolut

```
Set Variable: " /hauptClip/hauptUnterClip/unterClip2/:var1" = 0
```

Relativ

```
Set Variable: " ../unterClip2/:var1" = 0
```

Flash 5
Absolut

```
_root.hauptClip.hauptUnterClip.unterClip2.var1 = 0;
```

Relativ

```
_parent.unterClip2.var1 = 0;
```

Von `unterClip3` aus soll `hauptClip` gestoppt und `hauptClip` um 10 Grad gedreht werden:

Flash 4
Absolut

```
Begin Tell Target ("/hauptClip")
      Go to and Stop ("StopMarke")
End Tell Target
Set Property ("/hauptClip", Drehung) = GetProperty ("hauptClip",
_rotation) + 10
```

Relativ

```
Begin Tell Target ("../../")
      Go to and Stop ("StopMarke")
End Tell Target
Set Property ("../../", Drehung) = GetProperty ("../../",
_rotation) + 10
```

Flash 5
Absolut

```
_root.hauptClip.gotoAndStop ("StopMarke");
_root.hauptClip._rotation = _root.hauptClip._rotation +10;
```

oder

```
_root.hauptClip._rotation += 10;
```

Relativ

```
_parent._parent.gotoAndStop ("StopMarke");
_parent._parent._rotation +=10;
```

Der Array-Zugriffsoperator []

Mit dem Array-Zugriffsoperator [] ist es nicht nur möglich, auf die Elemente eines Arrays zuzugreifen, sondern auch die Flash 4-kompatiblen Funktionen wie `eval` zu vermeiden.

In unserem obigen Beispiel hatten wir in jedem der drei `unterClips` von `hauptUnterClip` jeweils fünf Variablen `var1` bis `var5` deklariert.

Alle `unterClips` sollen von der Hauptzeitleiste aus gestoppt und die Variablen `var1` bis `var5` in allen `unterClips` gesetzt werden:

Flash 4

```
Set Variable: "i" = 1
Loop While (i<4)
      Begin Tell Target ("hauptClip/hauptUnterClip/unterClip"&i)
            Go to and Stop ("StopMarke")
      End Tell Target
      Set Variable: "k" = 1
      Loop While (k<6)
            Set Variable: "hauptClip/hauptUnterClip/
unterClip"&i&"/:var"&k = k*i
            Set Variable: "k" = k+1
      End Loop
      Set Variable: "i" = i+1
End Loop
```

Flash 5

```
for (i=1; i<4; i++) {
    hauptClip.hauptUnterClip["unterClip"+i].gotoAndStop("Stop-
Marke");
    for (k=1; k<6; k++) {
        hauptClip.hauptUnterClip["unterClip"+i]["var"+k] = k*i;
    }
}
```

Im obigen Fall würde man in Flash 5 die Variablen `var1` bis `var5` als Array definieren und nicht als einzelne Variablen.

Jede Variable `var1` soll in den unterClips von der Hauptzeitleiste aus auf den Wert von `var5` gesetzt werden.

Flash 4

```
Set Variable: "i" = 1
Loop While (i<4)
      Set Variable: "hauptClip/hauptUnterClip/unterClip"&i&"/:var1"
= eval ("hauptClip/hauptUnterClip/unterClip"&i&"/:var5")
      Set Variable: "i" = i+1
End Loop
```

Flash 5

```
for (I=1; I<4; I++) {
      hauptClip.hauptUnterClip["unterClip"+i].var1 = hauptClip.
hauptUnterClip["unterClip"+i].var5;
}
```

Der Array-Zugriffsoperator wird wie folgt eingesetzt: [] ersetzt der Punkt an der Stelle, wo der zusammengesetzte Ausdruck beginnt.

Innerhalb von [] wird der zusammengesetzte Ausdruck angegeben. Die Teilausdrücke werden mit + zusammengesetzt. Wobei alle fixen Teilausdrücke in Anführungszeichen stehen. Auf den Operator [] folgt ein Punkt, und nach dem Punkt wird der Rest des Pfades angegeben, eine Methode aufgerufen oder eine Eigenschaft gesetzt.

Für den Fall, dass auf den Operator [] ein weiterer zusammengesetzter Ausdruck folgen sollte, wird statt eines Punktes ein weiterer Array-Zugriffsoperator benutzt.

Es ist somit möglich, komplett auf die Slash-Syntax zu verzichten und Flash 4-kompatible Funktionen zu vermeiden.

Der With-Operator

with ist ein wichtiger Operator, um effizient mit Objekten umzugehen. Statt

```
Objekt.Anweisung1();
Objekt.Anweisung2();
…
Objekt.AnweisungN();
Objekt._Eigenschaft = Wert;
```

benutzen wir die kompaktere Syntax

```
with (Objekt) {
      Anweisung1();
      Anweisung2();
   ...
      AnweisungN();
      _Eigenschaft = Wert;
}
```

Mit **with** kann man auch die veraltete Tell Target-Aktion ersetzen:

unterClips1 soll von der Hauptzeitleiste aus gestoppt und um zehn Pixel verschoben werden.

Flash 4

```
Begin Tell Target ("hauptClip/hauptUnterClip/unterClip1")
      Go to and Stop ("StopMarke")
End Tell Target
Set Property ("hauptClip/hauptUnterClip/unterClip1", X-Position) =
GetProperty ("hauptClip/hauptUnterClip/unterClip1", _x) +10
Set Property ("hauptClip/hauptUnterClip/unterClip1", Y-Position) =
GetProperty ("hauptClip/hauptUnterClip/unterClip1", _y) +10
```

Flash 5
Kompatible Syntax

```
tellTarget ("hauptClip/hauptUnterClip/unterClip1") {
      gotoAndStop ("StopMarke");
```

```
      _x += 10;
      _y += 10;
}
```

with-Syntax

```
with (hauptClip/hauptUnterClip/unterClip1) {
      gotoAndStop ("StopMarke");
      _x += 10;
      _y += 10;
}
```

Es gibt auch noch die Möglichkeit, `with` zu verschachteln:

```
with (hauptClip.hauptUnterClip) {
      with (unterClip1) {
            gotoAndStop("StopMarke");
      }
      with (unterClip2) {
            gotoAndStop("StopMarke");
      }
}
```

Selbstdefinierte Objekte erstellen

Flash 5 bietet uns die Möglichkeit, selbst Objekte zu definieren und von ihnen Instanzen abzuleiten. Dies eröffnet uns unter anderem die Möglichkeit, unseren Code strukturierter, kompakter, übersichtlicher und lesbarer zu gestalten (der Code ist nicht mehr im Projekt überall verstreut).

Da selbst definierte Objekte keine sichtbaren Objekte sind, können sie nicht auf die Bühne gezogen werden; deshalb benötigen wir einen Operator, der eine Instanz erzeugt. Wir benutzen den `new`-Operator, um ein neues Objekt durch die Konstruktorfunktion des Objekts zu erzeugen. Der `new`-Operator entspricht dem Platzieren eines MovieClip-Objekts auf der Bühne.

Die Eigenschaften, die mit einem Anfangswert initialisiert werden müssen, werden der Konstruktorfunktion des Objekts als Parameter übergeben. Wenn die Instanz eines Objekts mit der Konstruktorfunktion des Objekts und dem `new`-Operator erzeugt wird, übergibt Flash das ver-

steckte Argument `this`. `this` ist eine Referenz auf das neu erzeugte Objekt.

Innerhalb des Objekts ermöglicht `this` einen Bezug auf das erzeugte Objekt. Wenn wir z. B. einen MovieClip auf die Bühne ziehen, wird ebenfalls das versteckte Argument `this` mit übergeben. Damit ist es uns möglich, innerhalb eines MovieClips mit `this.stop();` den MovieClip anzuhalten, da `this` eine Referenz auf den MovieClip ist.

Wer rein objektorientiert programmieren möchte, verzichtet daher auf die so genannten Basisaktionen im Aktionsregister. Statt über Umwegen mit der Basisaktion `Stop();` den Clip anzuhalten, rufen wir direkt die Methode `stop()` des MovieClips mit `this.stop()` auf. Zu jeder Basisaktion existiert eine direkte Methode, die über `this` aufgerufen wird.

Zur Information: `this` war in Flash 4 `".."` . Mit `setProperty ("..", _visible) = false` konnte man den MovieClip unsichtbar machen.

In Flash 5 würden wir es mit `setProperty (this, _visible, false);` oder `mit this._visible = false;` versuchen.

Wir beginnen, zum Verständnis ein einfaches Objekt zu erstellen.

Wir definieren die Konstruktorfunktion des Objekts `ObjBeispiel`. Der Objektname wird aus dem Namen der Konstruktorfunktion abgeleitet.

```
function ObjBeispiel (Parameter1,  Parameter2) {
```

Das sind die Eigenschaften des Objekts `ObjBeispiel`. Diese Eigenschaften sind Variablen, die Eigenschaften dieses Objekts speichern. Jede Eigenschaft muss mit einem Wert initialisiert werden. Das ist die Aufgabe eines Konstruktors.

```
//Initialisierung mit dem Wert von Parameter1
    this.Operand1 = Parameter1;
//Initialisierung mit dem Wert von Parameter2
    this.Operand2 = Parameter2;
//Initialisierung mit dem Wert 0
    this.OpErgebnis = 0;
```

Jetzt werden die Methoden des Objekts definiert. Methoden führen mit den Eigenschaften des Objekts Aktionen durch und erfüllen bestimmte Funktionen.

```
    this.Methode1 = function () {
```

Die folgende Methode soll z. B. die Eigenschaften **Operand1** und **Operand2** addieren und zurückgeben.

```
    return this.Operand1 + this.Operand2;
}
this.Methode2 = function () {
```

Diese Methode soll z. B. die Eigenschaften **Operand1** und **Operand2** multiplizieren und das Ergebnis in die Eigenschaft **OpErgebnis** schreiben.

```
    this.OpErgebnis = this.Operand1 * this.Operand2;
}
```

Es ist ebenfalls möglich, einen MovieClip zur Eigenschaft eines selbst definierten Objekts zu machen.

Wir können z. B. aus den Methoden dieses Objekts heraus mit **this.MC.stop()** den MovieClip **_root.BeispielMovieClip.ClipUnterMC1.ClipUnterMC2** stoppen, anstatt **_root.BeispielMovieClip.ClipUnterMC1.ClipUnterMC2.stop();) zu** schreiben.

Und aus der Zeitleiste von **_root.BeispielMovieClip.ClipUnterMC1.ClipUnterMC2** können wir mit **this.Obj.Methode2()** nun die **Methode2** dieses Objekts aufrufen oder mit **this.Obj.Operand1** den Wert der Eigenschaft **Operand1** verändern.

```
    this.MC = _root.BeispielMovieClip.ClipUnterMC1.ClipUnterMC2;
    _root.BeispielMovieClip.ClipUnterMC1.ClipUnterMC2.Obj = this;
}
```

Jetzt erzeugen wir zwei neue Instanzen des Objekts **ObjBeispiel** und spielen etwas damit herum.

```
myObj1 = new ObjBeispiel(1,2);
myObj2 = new ObjBeispiel(5,10);
```

Jetzt haben wir zwei Instanzen des Typs **ObjBeispiel**.

myObj1.Operand1 hat jetzt den Wert 1.
myObj1.Operand2 hat jetzt den Wert 2.
myObj1.OpErgebnis hat jetzt den Wert 0.

Und:

myObj2.Operand1 hat jetzt den Wert 5.
myObj2.Operand2 hat jetzt den Wert 10.
myObj2.OpErgebnis hat jetzt den Wert 0.

Wir können jetzt mit unseren Instanzen arbeiten.

```
var myVar1 = myObj1.Operand2 * myObj2.Operand2;
```
myVar1 hat jetzt den Wert 20 (2*10).
```
var myVar2 = myObj1.Methode1();
```
myVar2 hat jetzt den Wert 3 (1+2).
```
var myVar3 = myObj2.Methode2();
```
myVar3 hat jetzt den Wert 15 (5+10).
```
myObj1.Methode2();
```
myObj1.OpErgebnis hat jetzt den Wert 1 (1*2).
```
myObj2.Methode2();
```
myObj2.OpErgebnis hat jetzt den Wert 50 (5*10).
```
myObj1.Operand1 = 10;
```
myObj1.Operand1 hat jetzt den Wert 10.

Wie man erkennt, hat jede Instanz ihre eigene Kopie der Eigenschaften und Methoden des **ObjBeispiel**-Objekts.

Ein weiteres Beispiel für die Arbeit mit selbst definierten Objekten bieten wir Ihnen im Teil »Spieleprogrammierung« mit dem Spiel »Coco-Catch« ab Seite 146.

Die Eigenschaft Prototype

Die obige Technik, Methoden zu definieren, ist etwas ungeschickt, denn bei jeder neuen Instanz muss der Code der Methoden in die neue Instanz kopiert werden. Deswegen wollen wir die Vererbung von Methoden etwas verfeinern.

Flash erzeugt zu jeder Funktion automatisch die Eigenschaft **prototype**. Beim Erzeugen eines neuen Objektes durch seine Konstruktorfunktion **myObj1 = new BeispielObj (1, 2);** werden alle Eigenschaften und Methoden der Eigenschaft **prototype** (Eigenschaft des Konstruktors) zu Eigenschaften und Methoden der Eigenschaft **__proto__** (Eigenschaft des neu erzeugten Objektes). Die Eigenschaft **myObj1.__proto__** erbt die Eigenschaften des **ObjBeispiel**-Objekts über **ObjBeispiel.prototype**.

Die Übergabe von Werten zwischen den Eigenschaften **__proto__** und **prototype** wird als Vererbung bezeichnet.

Wir definieren jetzt eine weitere Methode und weisen sie der **prototype**-Eigenschaft des **ObjBeispiel**-Objekts zu.

```
function() _Methode3_ () {

  return Math.pow(this.Operand1, this.Operand2);
}
```

Diese Methode soll z. B. die Eigenschaften **Operand1** und **Operand2** potenzieren und das Ergebnis zurückgeben.

Der folgende Aufruf hätte keine Wirkung und ist ein Fehler, da **_Methode3_** die Eigenschaften **Operand1** und **Operand2** nicht kennt.

```
var myVar4 = _Methode3_ ();
```

Wir machen **_Methode3_** zu einer Methode von **ObjBeispiel**:

```
ObjBeispiel.prototype.Methode3 = _Methode3_;
```

Jetzt ist **Methode3** eine Methode von **ObjBeispiel** und kennt deren Eigenschaften.

Wir erzeugen eine neue Instanz von **ObjBeispiel**:

```
myObj3 = new ObjBeispiel(2,10);
```

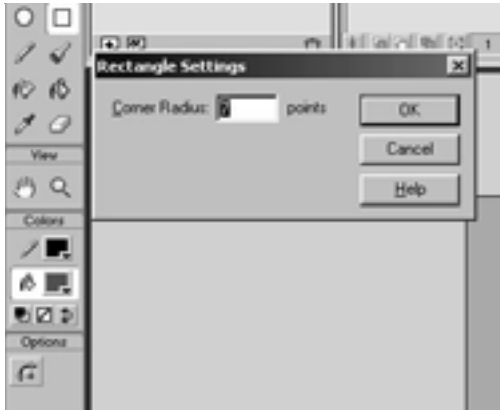
myObj3.__proto__ erbt von **ObjBeispiel.prototype** alle Eigenschaften, also auch **Methode3**.

Wenn wir jetzt **EineVariable = myObj3.Methode3 ();** aufrufen, sucht Flash in **myObj3** nach der **Methode3** und findet sie nicht. Daraufhin sucht Flash in **myObj3.__proto__**, findet die **Methode3** und ruft sie auf. **myVar4** hat jetzt den Wert 1024 (2 hoch 10).

```
var myVar4 = myObj3.Methode3 ();
```

Diese Technik ist die eleganteste und effektivste Art, Methoden zu definieren, da nur die Definition der Methoden in neue Instanzen kopiert werden muss und nicht der gesamte Code der Methoden.

▲ Abbildung 1
Das Endergebnis



▲ Abbildung 2
Abgerundete Ecken

Eigendynamik mit Zufallswerten erzeugen

In folgendem Beispiel zeigen wir Ihnen, wie Sie Filmsequenzen per ActionScript duplizieren und positionieren. Anschließend sollen die Filmsequenzen per Zufall ausgeblendet werden.

◀
Martin Fleck
www.action-script.com

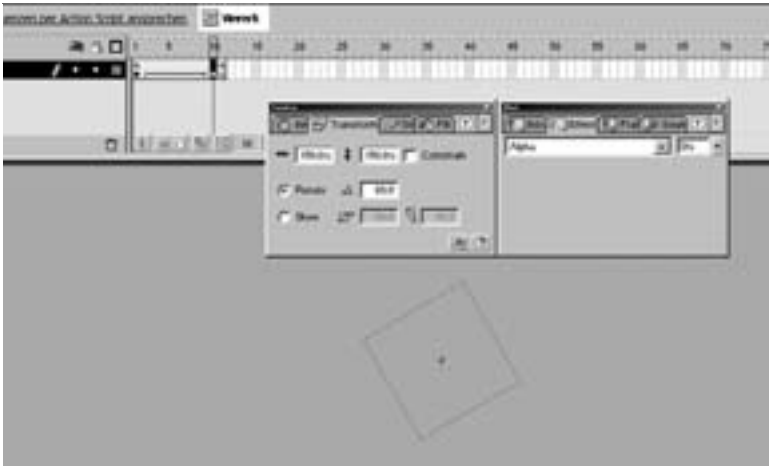
Vorbereitung

Verwenden Sie Einfügen·Neues Symbol (Strg+F8), um eine neue Filmsequenz (MovieClip) einzufügen. Im darauf folgenden Dialog Symboleigenschaften vergeben Sie als Namen »Viereck« und klicken auf OK.

Klicken Sie nun das Werkzeug Rechteck in der Palette Tools an (siehe Abb. 2). Mit der Option Eckradius bestimmen Sie die Intensität der Abrundungen. Als Beispiel verwenden Sie ein Quadrat, damit ein gleichmäßiger Bildaufbau entsteht. Erstellen Sie das Quadrat mit gedrückter ⇧-Taste.

Als Nächstes erstellen Sie das Endbild, indem Sie den gewünschten Frame markieren und ein Schlüsselbild einfügen (F6). Anschließend sollten Sie ein Bewegungstween hinzufügen. Öffnen Sie über Menüpunkt Fenster·Palette·Effekt und Transformieren. Selektieren Sie das Quadrat im Endbild, und skalieren Sie es auf 150 %. Geben Sie als Rotationswinkel 60° an. Wechseln Sie nun zur Effektepalette und wählen aus der Drop-down-Liste den Eintrag Alpha, setzen Sie als Wert 0 %.

Abbildung 3 ▶
Die Transformation im Endbild



Doppelklicken Sie auf das Startbild, und schreiben Sie `stop()`; danach schließen Sie den ActionScript-Editor. Fügen Sie im Frame nach dem Endbild ein leeres Schlüsselbild (F7) ein. Durch Doppelklick schreiben Sie wieder die Stopp-Aktion hinzu. Wechseln Sie zur Hauptebene zurück.

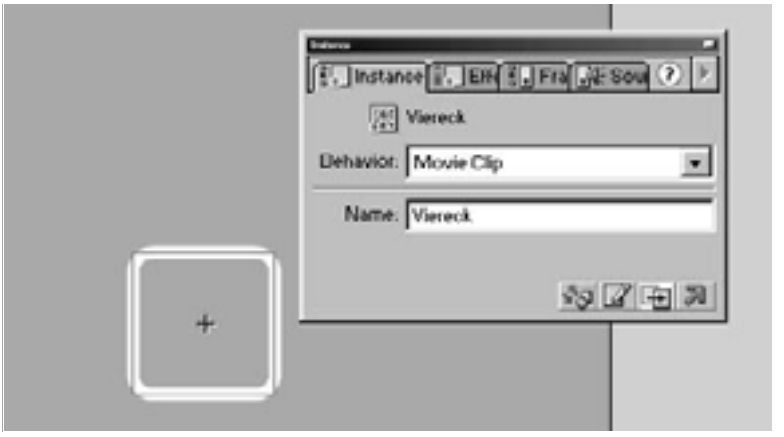
Öffnen Sie die Bibliothek des Films (Strg+L), und ziehen Sie die Filmsequenz »Viereck« aus der Bibliothek auf die Bühne. Die Positionierung der Filmsequenz ist nicht maßgebend, da sie zur Laufzeit ausgeblendet wird.

Öffnen Sie nun über das Menü Fenster·Palette·Instanz. Vergeben Sie als Namen »Viereck«. Nun können Sie per ActionScript die Filmsequenz ansprechen.

Filmsequenz duplizieren

Klicken Sie auf den Menüpunkt Einfügen·Neue Ebene. Vergeben Sie als Namen »Aktionen«. Nach Doppelklick auf das erste Schlüsselbild (leer) fügen Sie folgendes ActionScript ein:

```
1: Breite = 100;
2:
3: for (x=0; x<3; x++) {
4:     for (y=0; y<3; y++) {
5:
6:         Tiefe++;
```



◄ Abbildung 4
Den Instanznamen vergeben

```
7:       duplicateMovieClip ("/Viereck","Viereck" + x + y, Tiefe);
8:       _root["Viereck" + x + y]._x = 95 + Breite * x;
9:       _root["Viereck" + x + y]._y = 50 + Breite * y;
10 :
11:    }
12: }
13:
14: Viereck.gotoAndStop(11);
```

Erklärung nach Zeilen:

1: Abstand zu den anderen Vierecken in vertikaler und horizontaler Richtung (Wert in Pixel).

3: Schleife, die sich wiederholt, solange x kleiner als 3 ist. Der Anfangswert von der Zählervariable x ist 0.

4: Schleife, die sich wiederholt, solange y kleiner als 3 ist. Der Anfangswert von der Zählervariablen y ist 0.

6: Variable Tiefe wird bei jedem Durchlauf um 1 erhöht.

7: Erstellt eine Kopie einer Filmsequenz an der gleichen Position. Das erste Argument, das übergeben wird, ist der Zielpfad der zu duplizierenden Filmsequenz. Anschließend muss ein neuer Bezeichner für die aktuell duplizierte Filmsequenz vergeben werden, Sie erreichen dies durch das Anhängen der Variablen x und y an den String »Viereck«. Beispiel eines Bezeichners einer duplizierten Filmsequenz beim allerersten Durchlauf der beiden Schleifen: »Viereck00«.
Beide Zählervariablen (x und y) haben den Wert 0.

8: Nun wird der soeben duplizierten Filmsequenz die horizontale Position zugewiesen, diese entsteht durch das Multiplizieren des aktuellen x-Wertes mit der Breite.
9: Die vertikale Position wird ebenfalls der Filmsequenz zugewiesen.
11: Schleifenende x.
12: Schleifenende y.
13: In der Filmsequenz »Viereck« wird zum leeren Frame 11 gewechselt.

Fügen Sie im zweiten Frame ein weiteres Schlüsselbild (F6) hinzu, und durch Doppelklick öffnen Sie das ActionScript-Fenster:

```
1: x = Random(3);
2: y = Random(3);
3:
4: _root["Viereck" + x + y].gotoAndPlay(2);
5: _root["Viereck" + x + y]._name = "Viereck" + xy;
```

Erklärung nach Zeilen:
1: Der Variablen x wird ein Zufallswert zugewiesen, entweder 0, 1 oder 2.
2: Der Variablen y wird ein Zufallswert zugewiesen, entweder 0, 1 oder 2.
4: Nun wird eine Filmsequenz mit den Zufallswerten der Variablen x und y angesprochen. Damit startet die Animation der Filmsequenz, und das entsprechende Viereck wird ausgeblendet.
5: Der soeben gestarteten Filmsequenz wird ein neuer Name gegeben, um erneutes Aufrufen zu vermeiden.

Im dritten Frame fügen Sie wieder ein weiteres Schlüsselbild (F6) hinzu, und mit Doppelklick öffnen Sie das ActionScript-Fenster. Durch Schreiben von `gotoAndPlay (2);` wird zu Frame 2 gewechselt.

Ähnliche Effekte finden Sie auf folgenden Webseiten
www.derbauer.de
www.phong.com
www.foulds2000.freeserve.co.uk
www.mtv.co.uk

Bézierkurven in Flash

Mathematische Formeln nutzen

Das Beste an Flash ist doch die Möglichkeit, ActionScript einzusetzen. Dabei soll es nicht bei On (Press) bleiben. Man kann auch mathematische Formeln einsetzen, um z. B. die Bewegung der Symbole zu bestimmen.

▸ boblgum
www.mysterion.de

Damit kommen wir zu den Bézierkurven. Diese Kurven werden durch vier Punkte charakterisiert. Um den Verlauf der Kurve auszurechnen, brauchen wir X- und Y-Koordinaten von vier Punkten (P1 bis P4). Man setze die Koordinaten der Punkte ein, jeweils ein Mal für X und ein Mal für Y, und schon haben wir die Koordinaten für unsere Kurve. Damit sich die Koordinaten auch verändern und somit überhaupt eine Kurve entsteht, setzen wir eine Laufvariable »t« ein, die Werte von 0 bis 1 haben darf/kann. Wenn man möchte, dass die Kurve innerhalb der von vier Punkten abgegrenzten Fläche verläuft, darf »t« nicht größer 1 werden. Der Anfang der Kurve liegt dabei immer bei P1 und das Ende bei P4, wenn man genau nach der Formel vorgeht natürlich. Je weiter man von den Endpunkten (P1 und P4) der Kurve ist, desto dichter wird die Kurve und die Bewegung langsamer.

So viel Theorie dürfte wohl ausreichend sein. Gehen wir nun zu ActionScript über.

Aufbau des Films

Was wir brauchen:

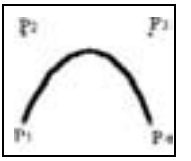
1. vier Mal den Punkt-MovieClip
2. ein Mal den Kurvenpunkt-MovieClip
3. ein Mal den Zeichner-MovieClip

$$Q(t) = (1-t)^3 * P_1 + 3t * (1-t)^2 * P_2 + 3t^2 * (1-t) * P_3 + t^3 * P_4$$

▲ **Abbildung 1**
Parameterdarstellung der Kurve

$$x = (1-t)^3 * x_1 + 3t * (1-t)^2 * x_2 + 3t^2 * (1-t) * x_3 + t^3 * x_4$$

▲ **Abbildung 2**
Kurvenformel für X-Werte



▲ **Abbildung 3**
Das Prinzip

Aufbau des Punkt-MovieClips

Als Erstes legen wir alle benötigten Ebenen an (von unten nach oben):

- ActionScript
- Zeichner
- Punkte

Dann zeichnen wir einen kleinen Kreis, der als einer der Punkte dienen wird. Den wandeln wir auch gleich in einen MovieClip um (MovieClip Punkt). Mehr brauchen wir auch nicht, weil man ja die MovieClips duplizieren kann. Hauptsache ist aber, dass jeder von ihnen einen anderen Instanznamen bekommt. Dann fügen wir noch einen Button in den MovieClip ein. Bevor wir aber die MovieClips duplizieren, setzen wir folgende Aktionen für den Button ein:

```
On (press) {
   _root.zeichner.gotoAndPlay(2);
   startDrag ("", true);
}
on (release, releaseOutside) {
   _root.zeichner.gotoAndStop(1);
   stopDrag ();
}
```

Hier geben wir ein, dass, wenn man auf einen der Kreise klickt und die Maustaste nicht loslässt, dann die Instanz `zeichner` zum Keyframe 2 gehen soll und abspielt. Sobald man die Maustaste loslässt, soll `zeichner` zum Keyframe 1 gehen und dort stehen bleiben. Diese Vorgehensweise spart wertvolle Systemressourcen, was später deutlicher wird.

Nun können wir den MovieClip auf der Bühne duplizieren (nicht in der Bibliothek!) und den einzelnen Kreisen Instanznamen vergeben (von Punkt 1 bis Punkt 4).

Einen Punkt brauchen wir noch. Und zwar den Punkt, aus dem sich die Kurve zusammensetzen wird. Wir zeichnen also einen kleinen Punkt (2 x 2px oder so), wandeln ihn in einen MovieClip um und geben ihm den Instanznamen »kurvenpunkt«. Den legen wir außerhalb der Bühne ab.

Weil wir ja schon etwas mehr als nur einen Punkt brauchen, um eine Kurve zu zeichnen, fügen wir in den ersten Keyframe der Ebene 1 (ActionScript) der Hauptzeitleiste folgendes ActionScript ein:

```
function zeichnen_x (t, x1, x2, x3, x4) {
   return (((1-t)*(1-t)*(1-t))*x1)+((3*t)*((1-t)*(1-
t))*x2)+(((3*(t*t))*(1-t))*x3)+((t*t*t)*x4);
};

function zeichnen_y (t, y1, y2, y3, y4) {
   return (((1-t)*(1-t)*(1-t))*y1)+((3*t)*((1-t)*(1-
t))*y2)+(((3*(t*t))*(1-t))*y3)+((t*t*t)*y4);
};

function zeichnen_kurve() {
   i = 1;
   do {
      _root[i]._x=zeichnen_x(i/100, punkt1._x, punkt2._x, punkt3._x,
punkt4._x);
      _root[i]._y=zeichnen_y(i/100, punkt1._y, punkt2._y, punkt3._y,
punkt4._y);
       i++;
   } while (i<100);
};

do {
   duplicateMovieClip (_root.kurvenpunkt, i, i);
   i++;
} while (i<100);
```

Mit diesen Anweisungen werden 100 Duplikate der Instanz `kurvenpunkt` angelegt. Jedes dieser Duplikate erhält als Instanznamen den Wert der Laufvariablen i und wird auf dem Level i abgelegt (nicht zu verwechseln

mit Ebenen in der Zeitleiste!). Außerdem definieren wir drei Funktionen, auf die wir dann zugreifen werden. Ich versuche es mir so anzugewöhnen, dass ich alle Funktionen auf der Hauptzeitleiste definiere. So weiß ich auch nach zwei Jahren, wo ich gucken muss, wenn ich den Film mal ändern will.

Die Funktionen (eine für die X-Werte und eine für die Y-Werte der Kurve) sind so definiert, dass sie nur einen gültigen Wert ausgeben, wenn man auch entsprechende Argumente an die Funktion übergibt. Das sind die Werte, die wir brauchen, um die Kurvenpunkte zu berechnen: X- und Y-Koordinaten der Punkte (von 1 bis 4) und die Laufvariable »t«. Das Gute an solch einer Strukturierung ist auch, dass jederzeit auch andere Objekte diese Funktionen nutzen können (was natürlich auch anders, aber doch nicht so gut geht), z. B. eine Kugel, die an der Kurve entlang läuft. Man könnte die drei Funktionen auch in einer zusammenfassen, was aber die Weiterverwendung der einzelnen Funktionen einschränkt.

Aufbau des Zeichner-MovieClips

Nun erstellen wir einen MovieClip, der unsere Kurve `zeichnen` wird. Dieser MovieClip besteht aus einer Ebene mit drei Keyframes.
Frame 1:

```
_root.zeichnen_kurve();
stop();
```

Frame 2:

```
_root.zeichnen_kurve();
```

Frame 3:

```
gotoAndPlay(_currentframe-1);
```

Wie schon oben geschrieben, wollen wir die Systemressourcen nicht unnötig belasten. Deswegen ist die letzte Aktion im Keyframe 1 auch eine Stopp-Aktion. Im zweiten Keyframe wird dieselbe Funktion wieder aufgerufen. Da wir den Film aber im dritten Keyframe zurück zum zweiten schicken und so eine Schleife erzeugen, wird unsere Kurve auch in Echtzeit gezeichnet, auch wenn wir die Punkte verschieben. Man könnte

auch ohne den `mcZeichner` auskommen, indem man in die Objektaktionen eines der Punkt-MovieClips Folgendes einfügt:

```
onClipEvent (mouseMove) {
   _root.zeichnen_kurve();
}
onClipEvent (mouseUp) {
   _root.zeichnen_kurve();
}
```

Diese Methode ist aber nicht so schön (wohl aber sparsamer in Bezug auf die Dateigröße), weil die Kurve dann nur nachgezeichnet wird, wenn die Maus bewegt bzw. losgelassen wird.

Nun sind wir auch schon fertig. Um Ihnen aber zu zeigen, was ich mit der Wiederverwendbarkeit der Funktionen gemeint habe, machen wir Folgendes: Wir lassen eine Kugel auf der Kurve entlang laufen.

Dazu legen wir einen MovieClip an (eine Kugel natürlich), und zwar mit zwei Ebenen und zwei Keyframes.
Frame 1:

```
step = (step > 0 and t < 1 or t <= 0) * 0.01 - (step < 0 or t >= 1) *
0.01;
t=t+ step;
_x=_root.zeichnen_x(t, _root.punkt1._x, _root.punkt2._x,
_root.punkt3._x, _root.punkt4._x);
_y=_root.zeichnen_y(t, _root.punkt1._y, _root.punkt2._y,
_root.punkt3._y, _root.punkt4._y);
```

Frame 2:

```
gotoAndPlay(_currentframe-1);
```

In Frame 1 benutzen wir die Funktionen aus der Hauptzeitleiste, um die Koordinaten zu ermitteln. Was für einige wohl auch nicht uninteressant sein könnte ist die Variable `div`. Diese Variable nenne ich »Pendelvariable«, weil sie Werte aus einem bestimmten Bereich und mit einem bestimmten Intervall haben kann.

Für diejenigen, die überhaupt nicht mit der ersten Zeile klarkommen:

Um diese Anweisung zu verstehen, muss man wissen, was **boolsche Variablen** bedeuten. Das sind Variablen, die entweder wahr oder falsch

sein können. Wenn die Variable wahr ist, kann man das (der PC tut das auch) auch als »gleich 1« interpretieren und falsch als »gleich 0«. Was passiert, wenn der Film abgespielt wird? Sobald der Film geladen wurde, wird `t` und `step` gleich 0 gesetzt, weil die Variable ja noch nicht deklariert wurde (ist eine Eigenart von Flash). So wird die erste Zeile gleich

```
step=1*0.01-0*0.01=0.01
```

sein, weil ja t <= 0 ist, was der IF-Abfrage in der ersten Klammer entspricht. Dann wird `t` um den `step`-Wert erhöht, bis `t` gleich 1 ist. Dann trifft die zweite Klammer zu und `step` wird als

```
step=0*0.01-1*0.01=-0.01
```

interpretiert. So pendelt es hin und her, daher der Name Pendelvariable. Will man nun die Geschwindigkeit der Bewegung erhöhen, muss man den `step`-Wert erhöhen. Also nicht 0.01, sondern etwas Größeres nach dem *-Zeichen einsetzen.

Nun sind wir aber endgültig fertig!

Beispiele für den Einsatz von ActionScript

Nach diesem theoretischen Einführungsteil kommen wir nun direkt zur Praxis, denn unserer Meinung nach lernt man so am besten.

Yugop-Menü in Flash 5

▶ Saban Ünlü
www.netTrek.de,
Stephan Fischer
www.phaetons.de

Im Juni 1999 machte die japanische Agentur MONO*crafts mit einem neuen Site-Interface Furore. Das Hauptmenü war als Laufband gestaltet und änderte abhängig von der Mausposition die Laufrichtung und Skalierung. Zurzeit wird die Site überarbeitet (wir sind gespannt, was sie dieses Mal aushecken), aber unter http://www.yugop.com/ver2 kann man die alte Version noch sehen.

Während man in Flash 4 noch mit Tricks arbeiten musste, um überhaupt die Position der Maus herauszufinden, bietet Flash 5 einige Funktionen, welche die Erstellung eines solchen Laufband-Menüs wesentlich erleichtern. (Die Skripte und Funktionen werde ich genauestens erklären, das Erstellen einer Filmsequenz und Vergeben von Instanznamen sollte allerdings geläufig sein und wird hier nicht näher erklärt.)

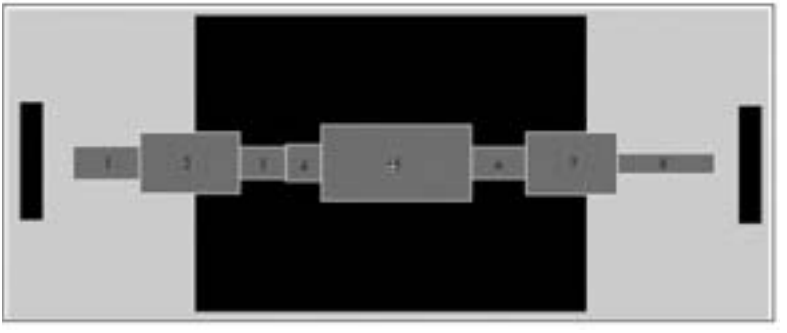
Das Prinzip

In unserem Beispiel haben wir acht Menüpunkte (es können aber auch beliebig viele sein), die alle als einzelne Filmsequenz erstellt wurden. Diese sollen wie ein Laufband von rechts nach links bzw. andersherum über den Bildschirm laufen.

Damit es nachher aussieht, als würde ein unendliches Band über den Bildschirm laufen, muss das Menü jeweils rechts und links über die Bühne herausragen, einige Menüpunkte sind also anfangs nicht oder nicht ganz zu sehen.

Warum die Menüs unterschiedlich groß sind (siehe Abbildung 2) und was es mit den beiden schwarzen Balken rechts und links auf sich hat, erkläre ich etwas später.

◂ **Abbildung 1**
Die Webseite
www.yugop.com



◂ **Abbildung 2**
Die Bühne mit den
Filmsequenzen

Im Wesentlichen benötigt man drei verschiedene ActionScripts, um die Funktion des Menüs zu verwirklichen.

1. Die Bewegungsgeschwindigkeit des Menüs muss abhängig vom X-Wert der Mausposition definiert werden. Hierzu werden alle Menüpunkte um eine bestimmte Anzahl Pixel nach links oder rechts verschoben.
2. Das Menü soll, abhängig von dem Y-Wert der Mausposition, skaliert werden. Damit nicht alle Filmsequenzen einzeln skaliert werden müssen, liegen alle Menüpunkte noch einmal in einer übergeordneten Filmsequenz.
3. Sobald ein Menüpunkt rechts oder links aus der Bühne herausläuft, soll dieser an das jeweilige andere Ende des Menüs positioniert werden. Hier kommen die beiden schwarzen Balken ins Spiel.

Die Filmsequenzen

Folgende Filmsequenzen werden benötigt:

1. Als Erstes benötigen wir die einzelnen Menüpunkte. Die Gestaltung bleibt vollständig Ihnen überlassen, und Sie können ganz normale Schaltflächen einbauen, wie bei Standardmenüs auch. Es muss nur jeder Menüpunkt als einzelnes Filmsequenzsymbol erstellt und den Instanzen auf der Bühne einheitliche Namen gegeben werden, damit man sie referenzieren kann. Wir haben in unserem Fall die Instanznamen »mc1« bis »mc8« vergeben. Wie schon erwähnt, können Sie eine beliebige Anzahl von Menüpunkten (Symbolen) verwenden.
2. Das Symbol »hit test« besteht aus einem senkrechten schwarzen Balken. Hiervon werden zwei Instanzen links und rechts von der Bühne positioniert und entsprechend mit den Instanznamen `rechts` und `links` versehen, damit wir sie später referenzieren können. Sie dienen dazu, um über eine Kollisionsabfrage die zu verschiebenden Menüpunkte herauszufinden.
3. Wenn Sie alle Symbole auf der Bühne platziert haben, markieren Sie bitte alle und erstellen daraus über Einfügen·In Symbol konvertieren ein neues Symbol. Im Beispiel hat es den Instanznamen `dieMovieClips` und ermöglicht es, alle Menüpunkte mit einer Aktion zu skalieren.

Die Filmsequenzen liegen in der Ebene `dieMCs`, die sich über drei Frames erstreckt. Darüber sieht man noch die Ebene Action, in der in drei verschiedenen Schlüsselbildern die Skripte für die Bewegung und die Skalierung liegen. Das Skript für die Verschiebung der äußeren Menüpunkte liegt in den Menüsymbolen selbst.

Hier sind alle Skripte aufgelistet. Die Erklärung steht als Kommentar zwischen den einzelnen Skriptzeilen.

In Frame 1 der Zeitleiste (Ebene »Action«) werden ein paar **globale Variablen** definiert, hier können Sie die Anzahl der Menüfilmsequenzen angeben (»anzahlMCs«). Machen Sie das wie folgt:

```
anzahlMCs = 8;
```

Die Gesamtbreite des Menüs wird später automatisch errechnet!

```
weite_dieMovieClips = 0;
```

Dann folgt eine Schleife zur Errechnung der Gesamtbreite des Menüs, sie hat den Startwert 0.

```
for (index=0; index <= anzahlMCs; index++) {
   weite = _root.die_MCs["mc"+index]._width -1;
   weite_dieMovieClips += weite;
}
```

Die Schleife wird durchlaufen, bis **index = anzahlMCs** und der Schleifenzähler um 1 erhöht ist, damit er bei 1 anfängt, und dann wird die Breite der Filmsequenz »mc1« bis »mc8« errechnet. Das ergibt die Gesamtbreite aller Menüfilmsequenzen.

Die Gesamtbreite wird in der Variablen **/die_MCs:weite_dieMovieClips** gespeichert.

```
_root.die_MCs.weite_dieMovieClips = Math.round
(weite_dieMovieClips);
```

Dieses Skript befindet sich im zweiten Schlüsselbild der Hauptzeitleiste und regelt die **horizontale Bewegung** des Menüs.

Hier wird mit v = die Anzahl der Pixel festgelegt, um die das Menü verschoben werden soll. Befindet sich die Maus in der Mitte der Bühne,

```
if (_xmouse>310 && _xmouse<330) {
```

ist v = 0, also gibt es keine Bewegung.

```
    v = 0;
} else {
```

Befindet sich die Maus in der linken Hälfte,

```
    if (_xmouse<=310 && _xmouse>0) {
```

liegt v zwischen 38 und 1 (Bewegung nach rechts).

```
        v = int((_xmouse-310)/-8);
```

Befindet sich die Maus in der rechten Hälfte,

```
    } else if (_xmouse>=330 && _xmouse<640) {
```

liegt v zwischen –38 und –1 (Bewegung nach links).

```
        v = int((_xmouse-330)/-8);
    }
}
```

Setze die Position auf die aktuelle Position + v.
`i=1` und `i<=anz. i++` ist äquivalent zu `i=0 i<anz. i=i+1`.

```
for (i=1; i <= anzahlMCs; i++) {
   _root.die_MCs["mc"+i]._x += v;
}
```

In Schlüsselbild 3 der Hauptzeitleiste befindet sich das Skript für die von der Y-Position der Maus **abhängige Skalierung des Menüs**.
Befindet sich die Maus in der Mitte,

```
if (_ymouse >230 && _ymouse <250) {
```

ist die Skalierung = 0.

```
    s = 0;
} else {
```

`s` wird Sicherheitshalber auf 0 gesetzt, damit sich keine Werte addieren.

```
    s = 0;
```

Befindet sich die Maus in der oberen Hälfte und ist die Skalierung nicht größer als 175 %,

```
    if (_ymouse<=230 && _ymouse>0 && _root.die_MCs._xscale<175) {
```

ändert sich die Skalierung um ca. 0 bis 6 %.

```
        s = int((_ymouse-240)/-40);
```

Befindet sich die Maus in der unteren Hälfte und ist die Skalierung nicht kleiner als 75 %,

```
    } else if (_ymouse>=250 && _ymouse<480 &&
_root.die_MCs._xscale>75) {
```

ändert sich die Skalierung um ca. 0 bis –6 %.

```
        s = int((_ymouse-240)/-40);
    }
}
```

X-Skalierung um den Wert s verändern

```
_root.die_MCs._xscale += s;
```

Y-Skalierung um den Wert s verändern

```
_root.die_MCs._yscale += s;
```

Damit die Bewegung und Skalierung nicht nur ein Mal geprüft werden, werden Schlüsselbild 2 und 3 ständig wiederholt.

```
gotoAndPlay (2);
```

Damit das Menü nicht auf einer Seite verschwindet, muss je nach Bewegungsrichtung immer der äußerste **Menüpunkt auf die andere Seite des Menüs versetzt** werden. Dazu wird mit Hilfe der Symbole »hit Test« (die schwarzen Balken rechts und links) und einer Kollisionsabfrage geprüft, welcher Menüpunkt versetzt werden muss.

Das Skript wird auf jede Filmsequenz (hier mc1 bis mc8) als Object-Aktion gelegt (Filmsequenz auf der Bühne markieren und dann FENSTER · AKTIONEN anklicken) und ist bei jeder Filmsequenz gleich!

```
onClipEvent (enterFrame) {
```

Wenn diese Filmsequenz sich mit der Filmsequenz `links` überschneidet,

```
if (this.hitTest(_root.die_MCs.links)) {
```

versetzen Sie diese Filmsequenz um die Gesamtbreite des Menüs nach rechts.

```
this._x += _parent.weite_dieMovieClips;
    }
```

Wenn diese Filmsequenz sich mit der Filmsequenz `rechts` überschneidet,

```
if (this.hitTest(_root.die_MCs.rechts)) {
```

versetzen Sie diese Filmsequenz um die Gesamtbreite des Menüs nach links.

```
this._x -= _parent.weite_dieMovieClips;
    }
}
```

Das war schon alles! Wer Lust hat, kann noch ein wenig mit den Werten v und s spielen, um die Geschwindigkeit bzw. Skalierung zu beeinflussen.

Für weitere Ausführungen zur Mausverfolgung beachten Sie bitte auch das Kapitel »Mausverfolgung« ab Seite 626.

Starfield-Simulation in Flash 5

▶ Peter Krempl www.flashspiele.de

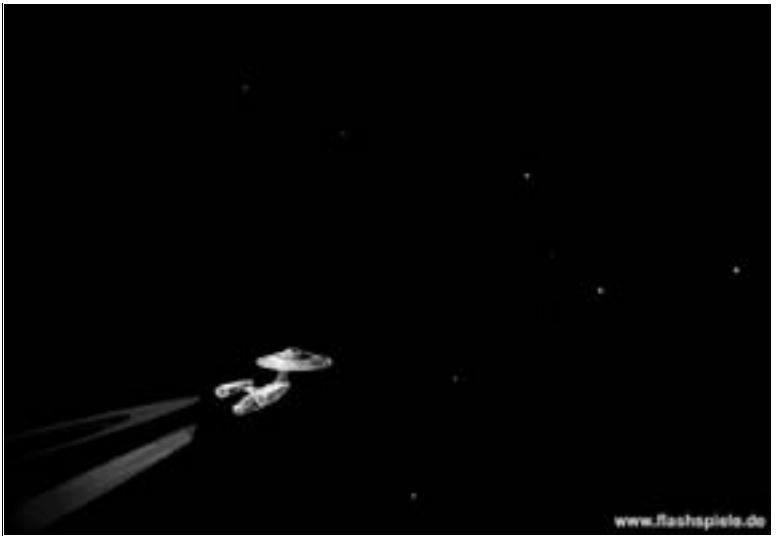
Dies hier ist eine kleine Spielerei mit `duplicateMovieClip`, um einen Raumflug durchs All zu simulieren, ähnlich dem bekannten Screensaver aus dem Hause Microsoft. Das Ergebnis eignet sich als Hintergrund zu einer einfachen Space-Schießerei.

Die Datei auf der CD-ROM nennt sich starfield.fla.

Nötig sind nur ein weißer Punkt in Form eines MovieClips und ein Raumschiff. Der Punkt MovieClip wird 35fach mit `duplicateMovieClip` in 35 Instanzen vervielfältigt.

```
for (var n=1; n<35; n++) {
 this.stern.duplicateMovieClip ("stern"+n,n);
}
play ();
```

Der MovieClip des Punktes bzw. des Sternes startet nun damit, dass er sich im ersten Frame einen Zufallsplatz mit negativen und positiven X- und Y-Werten sucht (30-Random(61)). Diese liegen alle unter 31 Pixel von der Mitte aus entfernt. Da der MovieClip selbst in einem weiteren Mo-



◀ Abbildung 3
Starfield-Simulation

vieClip liegt (wichtig), liegt sein Mittelpunkt nicht in der Mitte der Bühne, sondern am linken oberen Rand. Also bei `Y:0` und `X:0.` Der Alphawert wird auf 20 % gesetzt, die X-/Y-Skalierung auf 30 %.

```
this._x = 30-random(61);
this._y = 30-random(61);
stepx = random(1)/10+1.05;
stepy = random(1)/10+1.05;
this._xscale = 30;
this._yscale = 30;
this._alpha = 20;
this.gotoAndPlay (random(30));
```

Die Laufzeit zum nächsten aktiven Frame wird über einen Random-Befehl variiert, so dass nicht alle Sterninstanzen gleichzeitig starten.

Im nächsten aktiven Frame wird dem MovieClip die Geschwindigkeit in der Form gegeben, dass sein `_x` und `_y` mit einer Zufallszahl >1 multipliziert werden. Die Parameter `stepx` und `stepy` wurden in Frame 1 schon vorgegeben. Negative Zahlen bewegen sich nach unten, positive Werte nach oben. Damit entfernen sich alle Sterne von der Mitte aus weg. Durch die Multiplikation (an Stelle einer Addition) steigt das Tempo

schnell an: der Starfield-Effekt. Der Alphawert steigt auch, ebenso die Skalierung. Die Sterne werden größer, schneller und heller.

```
this._x *= stepx;
this._y *= stepy;
this._xscale *= 1.055;
this._yscale *= 1.055;
this._alpha *= 1.07;
```

Sobald die Sterne aus dem Blickfeld verschwunden oder zu groß geworden sind, wiederholen Sie die Prozedur von neuem.

```
if (this._x>300 || this._x<-300 || this._y>300 || this._y<-300) {
 this.gotoAndPlay (1);
}
if (this._xscale>300) {
 this.gotoAndPlay (1);
}
this.play ();
```

Mit 35 kleinen MovieClips entsteht so der Eindruck einer schnell vorbeiziehenden Galaxie. Fehlt noch ein Spiel im Vordergrund, z.B. Meteoritenschießen? Danke an Carlo Blatz für die Flash 5-Version.

Flash 5 Draw

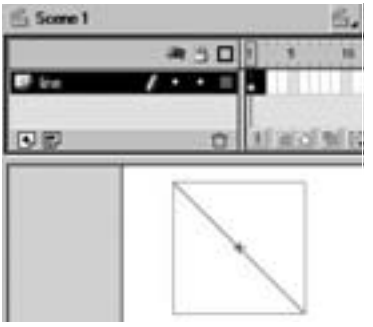
▶ Timo Meteling www.intabo.de, Björn Gromoll www.powerflasher.de

Zum Abschluss dieses Kapitels wollen wir Ihnen noch verraten, wie Sie mit Flash und ein wenig ActionScript ein vielseitiges Malprogramm erstellen können.

Natürlich kann man mit dieser Routine noch viele andere kreative Ideen verwirklichen, z.B. kann man mehrere Farben einsetzen oder farbige Hintergründe verwenden, z.B. eine Steinwand.

Man könnte Flash Draw auch benutzen, um ein Gästebuch zu programmieren. Das Gästebuch würde die Einträge der Nutzer dann in Echtzeit nachmalen!

Vielleicht ist das Netz bald voll von netten kleinen Grafiken, die sich nach dem Besuch der Seite direkt wieder löschen.



◀ Abbildung 4
Linie im 45°-Winkel

Ablauf

Fangen wir mit dem Aufbau des Flash-Files an: Dazu brauchen wir eine Szene und zwei Bilder in der Hauptzeitleiste. Die Framerate stellen wir auf 50 und die Bühnengröße auf ca. 700 x 400 Pixel ein.

Danach erstellen wir eine Filmsequenz mit dem Namen »line«. In diesen Film malen wir mit dem Linienzeichner eine Linie, die vom Ursprung aus in einem Winkel von 45° nach rechts unten läuft (benutzen Sie dazu die ⇧-Taste, damit die Linie auch wirklich exakt gerade läuft). Die Länge der Linie ist unerheblich, wichtig ist nur, dass der Winkel von 45° beibehalten wird.

Die Linie wird später unser Malstrich. Die dazugehörige Ebene können Sie deshalb auch genau so benennen.

Jetzt wird es ein bisschen schwieriger. Wir kommen nämlich zum ActionScript-Teil. Erzeugen Sie nun eine eigene Ebene, und nennen Sie diese »actions«. Wir brauchen zwei Bilder mit einem leeren Bild (F6). Achten Sie auch darauf, dass die untere Ebene »line« auch auf beiden Bildern sichtbar ist.

Nun benötigen wir noch einen Controller, der das Aktivieren unseres Zeichenstiftes übernimmt. Hierfür erstellen wir eine weitere Filmsequenz mit Namen Controller. Diese Filmsequenz kann an einem beliebigen Ort platziert werden, da sie später nur ActionScript enthalten wird.

Es handelt sich um einen onClip-Controller, der seine Aktion nur dann ausführt, wenn der Controller auch auf der Bühne ist. Die Bedingung ist `mouseDown` bzw. `mouseUp`.

Wir klicken nun mit der rechten Maustaste auf den Controller und wählen Aktionen aus (Alternative: Doppelklick + Alt-Taste).

Schauen wir uns jetzt das Skript im Controller an.

Controller

```
onClipEvent (mouseDown) {
    _parent.mouse = 1;
    _parent.x1 = 0;
}
onClipEvent (mouseUp) {
    _parent.mouse = 0;
}
```

Hier setzt das Controller-Skript die globale `mouse`-Variable entweder auf 1 oder 0. Zudem wird die Koordinatenvariable x1 auf 0 gesetzt, um dem Hauptskript einen Neubeginn der Skalierung unserer Linie mitzuteilen.

In das Bild der Hauptzeitliste setzen wir den Löwenanteil des Action-Scripts:

Wenn die Maus gedrückt wird, dann beginne mit diesem Skript:

```
if (mouse == 1) {
   if (x1 == 0) {
      x1 = (_xmouse);
      y1 = (_ymouse);
```

Ist die Maus zum ersten Mal gedrückt, setze x1 und y1 auf den Wert der Mauskoordinaten:

```
   } else {
      x0 = (_xmouse);
      y0 = (_ymouse);
```

Damit das Zeichnen von kaum sichtbaren Linien verhindert werden kann, wird ein Limbus aus den Deltawerten errechnet. Ist also der Abstand zwischen dem Bewegungsanfang und dem Bewegungsende der Maus kleiner oder gleich fünf Pixel, dann wird die Linie nicht erzeugt (das hat Gründe der Performance).

```
      if (((x0-x1)*(x0-x1))>5 || ((y0-y1)*(y0-y1))>5) {
```

n1 ist der Linienzähler. Er wird um 1 erhöht:

```
         n1++;
```

Erstelle eine Kopie aus dem Film »line« auf der Tiefe n1:

```
        line.duplicateMovieClip(["line"+n1], n1);
```

Position und Vektor der neuen Linie werden hier festgelegt:

```
        this["line"+n1]._x = x1;
        this["line"+n1]._y = y1;
  this["line"+n1]._xscale = x0-x1;
  this["line"+n1]._yscale = y0-y1;
```

Variablen werden übertragen, um den nächsten Deltawert errechnen zu können:

```
        x1 = x0;
        y1 = y0;
      }
   }
}
```

Und das war es schon! Danke an Carlo Blatz für die Flash 5-Version.

Smart-Filmsequenzen

Clever und Smart

Wir demonstrieren, wie man mit Smart-Filmsequenzen komplexe Menüs mit wenigen Mausklicks ändert und adaptierbar macht.

▸ Saban Ünlü

Smart-Filmsequenzen sind im Grunde »nur« Filmsequenzen im klassischen Sinne, allerdings mit einem Unterschied, der die Smart-Filmsequenzen oder Smart Clips zu einem der besten neuen Features von Flash 5 macht.

Man bindet in Smart-Filmsequenzen Aktionen ein und beeinflusst diese über vordefinierte Parameter, d.h. wenn Sie mehrere Instanzen einer Smart-Filmsequenz auf der Bühne haben, so kann jede Instanz eine andere Aktion ausführen, falls auch die Parameter unterschiedlich sind.

Nun ein Beispiel. Erstellen Sie ein dynamisches Textfeld, und weisen Sie ihm einen Variablennamen zu. Diese konvertieren Sie jetzt zu einer Filmsequenz und nehmen sie von der Bühne. In der Bibliothek führen Sie auf dieser Filmsequenz einen rechten Mausklick aus und klicken auf Sequenzparameter definieren.

Hier erscheint nun ein Parameterfenster, in dem Sie Variablen definieren können. Klicken Sie oben links auf Plus, und geben Sie in der neu erzeugten Zeile für Namen denselben Variablennamen, den Sie schon bei der Erstellung der Filmsequenz verwendet haben, ein. Den Rest belassen Sie so und beenden das Fenster mit OK.

Nun ziehen Sie die Filmsequenz aus der Bibliothek auf die Bühne und klicken hier mit der rechten Maustaste darauf. Achten Sie darauf, dass dabei die Filmsequenz markiert ist. Nun auf Bedienfelder·Sequenzparameter klicken. Sie sehen, dass fast das identische Fenster wie soeben geöffnet wird. Hier tragen Sie für die von Ihnen definierte Variable einen Wert ein, eine beliebige Zeichenfolge. Wichtig: Klicken Sie anschließend in einen leeren Bereich der Bühne, damit der Parameter übernommen wird. Wenn Sie den Film nun testen, sehen Sie, dass in dem eigentlich leeren Textfeld jetzt der zuvor vergebene Variablenwert zu sehen ist.

Was hat man davon?

Sie können diese Filmsequenz nun aus dieser Bibliothek auf verschiedene neue Projektbühnen legen und durch Einstellung der Sequenzparameter den Ausdruck des Textfeldes einfach bedienen. Sie haben nun also eine Vorlage erstellt.

netTrekMenu

Öffnen Sie die **netTrekMenu.fla** als Bibliothek und ziehen Sie aus dieser nun die Filmsequenz `netTrekMenu` auf ein Projekt Ihrer Wahl. Nun stellen Sie die Sequenzparameter ein.

Mit `DownOrUp` kann man definieren, wie sich das Menü aufbauen soll, nach unten oder nach oben.

In die Liste der Hauptpunkte gibt man die Menüpunktbezeichnungen für die Hauptpunkte ein. In die Liste Unterpunkte gibt man die Unterpunktbezeichnungen und die zugehörige Action ein. Beachten Sie hierbei folgende Syntax:

```
1*Unterpunkt1#1 *_root.stop()
```

`1*` gibt an, dass sich diese Unterpunktbezeichnung auf Hauptpunkt 1 bezieht, entsprechend bezieht sich `2*` auf den zweiten Hauptpunkt etc. Dem nächsten Stern ***** folgt die Action, die bei Klick auf diesem Unterpunkt ausgeführt werden soll. Dabei bitte darauf achten, dass dem Stern kein Leerzeichen folgt und dass vor dem Stern keine Zahl steht.

WICHTIG: Immer die Dot-Syntax verwenden und die Action immer im absoluten Pfad bezüglich eines MovieClips schreiben (auch beim `_root`.) Zurzeit funktioniert:

```
stop()
play()
gotoAndStop( frame )
gotoAndPlay( frame )
```

`frame` ist ein ganzzahliger Wert, der die Framezahl wiedergibt.

Nachdem diese Einstellungen erfolgt sind, baut sich das Menü so auf, wie Sie es soeben definiert haben. Haben Sie schon einmal ein Menü zur Steuerung eines Flash-Projektes erstellt?

Wie funktioniert meine Smart-Filmsequenz?

Die Filmsequenz beginnt damit, dass kontrolliert wird, ob es ein Pull-down oder -up werden soll:

```
if (DownOrUp.toUpperCase() == "PULLUPMENU") {
    UpDown = -1;
    tiefe = 1;
} else {
    UpDown = 1;
    tiefe = 500;
}
```

Entsprechend werden die Variablen `DownOrUp` und `tiefe` gesetzt. Diese Variablen werden benötigt, um später bei `duplicateMovieClip` die Tiefe definieren zu können.

So, nun muss sich das Menü aufbauen. Es müssen Arrays erzeugt werden, welche die Hauptmenübezeichnungen, die Untermenübezeichnungen und die Aktionen verwalten.

Das ActionScript ist objektbezogen und liegt in der Filmsequenz »MenuKomplett«:

```
onClipEvent (load) {
   index5 = 100;
   for (index=0; index<=_parent.Hauptpunkte.length; index++) {
      if (index<_parent.Hauptpunkte.length) {
   menuAufbau.duplicateMovieClip("HPkt"+(index+1),
(_parent.tiefe+index));
   this["HPkt"+(index+1)]._x = this._x+(index*(menuAuf-
bau.MenuPkt.grundFlaeche._width-1.3));
   this["HPkt"+(index+1)].MenuPkt.MenuPktText.menuPKT =
_parent.Hauptpunkte[index];
   this["HPkt"+(index+1)].hauptPktNummer = index+1;
      }
      this["subZu"+index] = new Array();
      this["actionZu"+index] = new Array();
      for (index2=1; index2<=_parent.Unterpunkte.length; index2++) {
        if (_parent.Unterpunkte[index2-1].indexOf(index+"*") != -1) {
            this["subZu"+index].push(_parent.Unterpunkte[index2-1].
substring((_parent.Unterpunkte[index2-1].indexOf(this[index+"*"])
```

```
+3), (_parent.Unterpunkte[index2-1].lastIndexOf("*"))));
            this["actionZu"+index].push(_parent.Unterpunkte[index
2-1].substring((_parent.Unterpunkte[index2-1].lastIndexOf("*")+1),
_parent.Unterpunkte[index2-1].length));
         }
      }
      liste = this["subZu"+index];
      liste2 = this["actionZu"+index];
      for (index3=0; index3<this["subZu"+index].length; index3++) {
         index5+=_parent.UpDown*(-1);
         subAufbau.duplicateMovieClip("Sub"+index+"_"+(index3+1),
index5);
         this["Sub"+index+"_"+(index3+1)]._x = this["HPkt"+index]._x;
         this["Sub"+index+"_"+(index3+1)]._y = menuAufbau.MenuPkt.
grundFlaeche._y+(index3+1)*(_parent.UpDown)*
(menuAufbau.MenuPkt.grundFlaeche._height-1.3);
         this["Sub"+index+"_"+(index3+1)]._visible = 0;
         this["Sub"+index+"_"+(index3+1)].hauptPktNummer = index;
         this["Sub"+index+"_"+(index3+1)].subNummer = index3+1;
         this["Sub"+index+"_"+(index3+1)].befehl = liste2[index3];
         this["Sub"+index+"_"+(index3+1)].MenuPkt.MenuPktText.
menuPKT = liste[index3];
      }
   }
   menuAufbau._visible = 0;
   subAufbau._visible = 0;
}
```

Haben Sie das auf Anhieb verstanden?

Wir sehen hier drei ineinander verknüpfte Schleifen: Die äußere Schleife analysiert die Anzahl der Hauptpunkte, die bei den Sequenzparametern eingetragen wurden und dupliziert entsprechend oft die Vorlage `menuAufbau`.

Die neuen Filmsequenzen werden nun relativ zur Filmsequenz `grundFlaeche` positioniert. Hierbei wird die Y-Koordinate übergangen, weil diese Koordinate der Vorlage entspricht.

Weiterhin wird den Filmsequenzen durch die Variable `hauptPktNummer` gesagt, welcher Hauptpunkt es ist, und durch `menuPkt`, welcher Text in das dynamische Textfeld in dieser Filmsequenz eingetragen werden soll.

In der ersten inneren Schleife werden zwei neue Arrays erzeugt, `subzu...` und `actionzu...`.

Die Elemente von `subzu...` sind die Unterpunktbezeichnungen und werden erzeugt, indem man die Elemente aus `Unterpunkte` (Sequenzparameter) zwischen den beiden Sternen * ausschneidet. In `actionzu...` ist entsprechend der Rest, der den Befehl wiedergibt, der auf den Unterpunkt mit Klick ausgeführt werden soll.

In der zweiten inneren Schleife wird `subAufbau` dupliziert, wodurch die Unterpunkte erzeugt werden. Dann werden diese positioniert und erhalten die Variablen `hauptPktNummer`, `subNummer` und `befehl`. Mit der Übergabe der Variable `menuPKT` wird der Inhalt des dynamischen Textfeldes und folglich die Unterpunktbezeichnung definiert. Die Unterpunkte werden hier auch auf unsichtbar gestellt, damit sie später, bei MouseOver über die Hauptpunkte, wieder sichtbar gemacht werden können.

Nachdem die Schleifen durchgearbeitet wurden, werden die Vorlagen unsichtbar gemacht. Diese werden jetzt nicht mehr benötigt. Auch die Vorlagen haben ein objektbezogenes ActionScript, das die Befehle für die Haupt- und Unterpunkte definiert.

In »Hauptpunkte«

```
onClipEvent (enterFrame) {
   neueHintergrundFarbe = new Color(MenuPkt.grundFlaeche);
   if (MenuPkt.grundFlaeche.hitTest(_root._xmouse, _root._ymouse,
flase)) {
      neueHintergrundFarbe.setRGB(0xaaaaaa);
      _parent["warte"+hauptPktNummer] = getTimer ();
   } else {
      neueHintergrundFarbe.setRGB(0xcccccc);
      if (getTimer ()  > (_parent["warte"+hauptPktNummer]+200)) {
        for (index=1; index<=_parent["subZu"+hauptPktNummer].length;
index++) {
            if (_parent["Sub"+hauptPktNummer+"_"+index].
_visible = 0) {
               break;
            }
      _parent["Sub"+hauptPktNummer+"_"+index]._visible = 0;
         }
      }
```

```
   }
}
```

Beim `MouseOver` wird über `hitTest` kontrolliert, ob sich die Maus noch über dem Menupunkt befindet. Falls ja, sollen die Unterpunkte sichtbar gemacht werden, und auch ein Farbwechsel der `grundEbene` soll ausgeführt werden.

Weiterhin wird eine Variable `warte...` definiert, mit der man später kontrollieren kann, ob die Maus sich noch auf dem Hauptpunkt bzw. Unterpunkt befindet oder nicht. Schließlich soll sich das Menü wieder einfahren, wenn sich die Maus nicht mehr auf den Punkten befindet.

In »Unterpunkte«

```
onClipEvent (mouseDown) {
   if ((MenuPkt.grundFlaeche.hitTest(_root._xmouse, _root._ymouse,
flase)) && (_visible)) {
      _parent["warte"+hauptPktNummer] = getTimer ();
      _parent.actionAusfuehren(befehl);
   }
}
onClipEvent (enterFrame) {
    neueHintergrundFarbe = new Color(MenuPkt.grundFlaeche);
    if (MenuPkt.grundFlaeche.hitTest(_root._xmouse, _root._ymouse,
flase)) {
        neueHintergrundFarbe.setRGB(0xaaaaaa);
        _parent["warte"+hauptPktNummer] = getTimer ();
        for (index=1; index<=_parent
["subZu"+hauptPktNummer].length; index++) {
            if (_parent["Sub"+hauptPktNummer+"_"+index]._visible) {
                break;
            }
            _parent["Sub"+hauptPktNummer+"_"+index]._visible = 1;
        }
    } else {
        neueHintergrundFarbe.setRGB(0xcccccc);
    }
}
```

Hier sehen wir, dass beim Mausklick die Funktion **actionAusführen** gestartet wird, die auf derselben Ebene wie die Vorlagen liegen. Auch hier wird die Variable **warte...** gesetzt. Nun wird geprüft, ob sich die Maus auf einem der Unterpunkte befindet. Falls ja, wird auch hier ein Farbwechsel der **grundEbene** ausgeführt und die Variable **warte...** gesetzt. Wenn nicht, wird geprüft, wie lange sich die Maus nicht mehr im Unterpunkt befindet. Dies macht man mit der Variable **warte...** und entsprechend werden die Unterpunkte wieder unsichtbar gemacht.

Die Funktion

```
function actionAusfuehren (action) {
   dummy = action.toUpperCase();
   if (dummy.indexOf("STOP(") != -1) {
      if (action.indexOf("gotoAndStop(") != -1) {
         tellTarget ( action.substring(0, (action.
indexOf("gotoAndStop("))-1)) {
            gotoAndStop (int(action.substring
((action.indexOf("(")+1), action.lastIndexOf(")"))));
         }
      } else {
         tellTarget ( action.substring(0, action.indexOf
("stop(")-1)) {
            stop ();
         }
      }
   }
   if (dummy.indexOf("PLAY(") != -1) {
      if (action.indexOf("gotoAndPlay(") != -1) {
         tellTarget ( action.substring(0, (action.indexOf
("gotoAndPlay("))-1)) {
            gotoAndPlay (int(action.substring
((action.indexOf("(")+1), action.lastIndexOf(")"))));
         }
      } else {
         tellTarget ( action.substring(0, action.indexOf("play(")-1)) {
            play ();
         }
      }
```

```
    }
}
```

Diese Funktion kontrolliert, welcher Pfad und Befehl ihr übergeben wurde, und führt diese entsprechend aus. Sie erinnern sich an die Variable `befehl`, die beim Aufruf der Funktion übergeben wird.

Dies geschieht, indem der übergebende Parameter durch geschicktes `subString` gegliedert und die einzelnen Elemente dann kontrolliert werden.

Ein Tipp: Zurzeit wird ein Font in das Menü eingebettet. Wenn Sie den dynamischen Textfeldern in der Smart-Filmsequenz Systemschriften zuordnen, so wird die Filmsequenz noch einmal wesentlich kleiner.

Eine benutzerdefinierte Oberfläche erstellen

Zur Modifizierung der Parameter der Smart-Filmsequenzen muss man aber nicht zwangsläufig das typische Bedienfeld SEQUENZPARAMETER benutzen. Wenn man es dem User einfach machen will, kann man auch eine benutzerdefinierte Oberfläche erstellen. Diese Oberfläche ist ein SWF, der die Parameter für die Smart-Filmsequenzen verwaltet. Durch diese neue benutzerfreundliche Oberfläche können selbst »Grafiker« programmieren.

Der SWF verwaltet die Variablen der Smart-Filmsequenzen mithilfe einer Filmsequenz. Diese muss den Instanznamen `XCH` tragen. Durch diese Benennung erreicht man eine Schnittstelle zwischen der benutzerdefinierten Oberfläche (SWF) und den Smart-Filmsequenzen; jede Variable, die in `XCH` definiert wird, ist ein Parameter für das ActionScript der Smart-Filmsequenzen.

Es sollte darauf geachtet werde, dass lediglich vordefinierte Variablen in der `XCH` modifiziert werden. Die dynamische Erstellung der Variablen oder verschachtelten Arrays kann zum Absturz von Flash 5 führen. Des Weiteren sollte man es vermeiden, komplexe Strukturen, insbesondere mehrere Filmsequenzen, in die Instanz XCH zu legen. Auch die Handhabung von Variablen sollte gut durchdacht sein, optimal wäre die Benutzung von überwiegend lokalen Variablen im gesamten Interface.

Das Beste wäre, wenn Sie `XCH` nur als Lagerort für Variablen nutzen würden!

Ein kleines Beispiel für die Smart-Filmsequenz »netTrekButtonV1.fla« und deren Interface »Interface_nTfB.swf« finden Sie auf der beiliegenden CD-ROM.

Gehen Sie wie folgt vor: Kopieren Sie beide Dateien in den Ordner »Bibliothek« Ihres Installationsverzeichnisses für Flash 5. Jetzt finden Sie unter FENSTER-ALLGEMEINE BIBLIOTHEKEN die Smart-Filmsequenz wieder. Betrachten Sie nun die Datei »anwendungs.fla«, sie zeigt Ihnen kurz, wie die Smart-Filmsequenz anzuwenden ist.

Auch die Datei zur Oberfläche »Interface_nTfB« liegt offen auf der CD-ROM. Auf eine detaillierte Erläuterung wird verzichtet, weil das Skripting gut zu lesen ist. Sollten dennoch Fragen offen bleiben, können Sie mich gern anschreiben.

Extensions in Flash

Zum Abschluss möchte ich erwähnen, dass es durch die neue Technologie der Smart-Filmsequenzen möglich ist, auch in Flash Extentions einzubringen. Dreamweaver-Benutzer sind dies bereits gewohnt. Sie finden die Extentions unter http://www.macromedia.com/exchange/flash. Zu beachten ist, das Sie den neuen Extention Manager dafür benötigen.

Der Extension Manager unterscheidet nicht zwischen der englischen und der deutschen Version von Flash. Die Extentions werden somit in englisch betitelten Ordnern, z. B. »Libraries«, abgelegt. Kopieren Sie diese einfach in die entsprechenden deutschen Ordner, z. B. »Bibliothek«, und Sie können die Extentions nutzen.

Spieleprogrammierung mit ActionScript

110 Grundlagen

111 Grundeinstellungen
111 Ebene Interface
112 Der Balken
115 Der Ball
121 Ebene Aktionen/ Einstellungen
122 Ebene Controller/Time
124 Ebene Text/Buttons

126 Flash-Huhn, Slotmaschine und Co.

126 Flash-Huhn
129 Minislotmaschine
130 Flash-Billard

134 Kollisionen, Highscores u. a.

134 Kollisionen
136 Highscore-Liste in Flash mit PHP4-Skript
143 Stoppuhr

146 Coco-Catch

147 Überblick über Objekte und Aktionen im Spiel
148 Reihenfolge der Implementierung des Spiels
150 Erstellung des Knilchs
152 Erstellung des Lochhindernisses
152 Erstellung des Krebshindernisses
154 Erstellung des Spinnenhindernisses
155 Erstellung der Affen und der Nüsse
156 Erstellung der Fahrzeuge
156 Implementierung der Objekte (der Code zur Steuerung des Spiels)
167 Methoden
186 Erzeugung der Instanzen obiger Objekte
188 Erstellung der Controll MovieClips zum Aufruf der Methoden der Objekte

Grundlagen

Einführung am Beispiel des Spiels Breakout

In diesem Kapitel erstellen Sie ein kleines Spiel und werden dabei im Umgang mit einigen elementaren ActionScript-Techniken, die zur Programmierung von Spielen nötig sind, vertraut gemacht.

▸ Johannes Pohl
www.2-serious.de

Flash 4 bot dem Spieleprogrammierer nur sehr eingeschränkte Möglichkeiten, Spielelemente zu realisieren. Ein wichtiges Stichwort ist beispielsweise die so genannte **Kollisionsabfrage** (die Methode, Begrenzungen verschiedener Objekte auszuwerten und festzustellen, ob diese sich an irgendeinem Punkt überschneiden oder überlappen). In Flash 4 konnte man diese Methode zwar umständlich emulieren, indem man durch mathematische Formeln die Werte und Positionen der äußeren Begrenzungsrahmen ermittelte und miteinander verglich, doch diese Operation benötigte viele Zeilen Code. Flash 5 ActionScript hingegen besitzt nun eine Methode namens **hitTest**, die all diese Schritte in einer Funktion vereint.

Des Weiteren wurde in Flash 4 die Implementierung der Tastatur ziemlich vernachlässigt. Mit dem neuen ActionScript wurde auch Abhilfe geschaffen. Somit steht nun dem Spieleentwickler höchstens noch mangelnde Kreativität im Weg.

Im Folgenden werden Sie anhand des kleinen (aber feinen) Spiels Breakout unter anderem lernen, diese grundlegenden Elemente einzusetzen.

Ziel des Spiels ist es, mit Hilfe eines über die Tastatur steuerbaren Balkens kleine Bälle in einem bestimmten Bereich zu halten. Je weniger Bälle diesen Bereich verlassen, desto mehr Punkte bekommt der Spieler. Am besten Sie machen sich selbst ein Bild des Spiels, das sich zusammen mit der Arbeitsdatei auf der beiliegenden CD-ROM befindet.

Grundaufbau

Das Spiel besteht aus programmiertechnischer Sicht im Wesentlichen aus zwei Hauptelementen, nämlich

- dem Balken und
- den Bällen.

Diese beiden Elemente sind Filmsequenzen, die sich selbst durch ihren zugewiesenen Code steuern.

Außerdem beinhaltet das Spiel noch folgende Unterelemente:

- Zeitanzeige
- Interface (Grafische Oberfläche, Buttons)

Am besten, Sie setzen sich gleich an Ihren Computer und fangen an mit den ...

Grundeinstellungen

In der Flash-Umgebung stellen Sie die Bildwiederholrate (Framerate) auf 20 fps (frames per second). Diese Einstellung stellt einen guten Kompromiss zwischen flüssigem Ablauf und nicht zu hoher Beanspruchung der Rechenleistung dar. Die Hintergrundfarbe setzen Sie am besten auf Schwarz, um später einen guten Kontrast zwischen Hintergrund und Bällen zu bekommen.

Legen Sie nun sechs Ebenen an.

Nennen Sie diese von oben nach unten:

- Aktionen/Einstellungen
- Balken
- Ball
- Controller/Time
- Text/Buttons
- Interface

Ebene Interface

Zunächst beginnen wir mit der letzten, untersten Ebene, dem Hintergrund des Spiels. Sie können diesen in einem externen Programm wie Adobe Photoshop entwerfen, wie ich es in beiliegendem Beispiel getan habe, doch das spielt in puncto Funktion keine Rolle. Es genügt auch vorerst ein provisorisch angelegter Hintergrund, um weiter programmieren

zu können, doch sollten Sie nach getaner Programmierarbeit auch am optischen Aspekt arbeiten. Dem Benutzer soll schließlich auch etwas fürs Auge geboten werden.

Zeichnen Sie also unter Zuhilfenahme des Rechtecktools den Hintergrund, wobei Sie darauf achten müssen, dass die unterste Ebene »Interface« dabei markiert ist. Wenn Sie mit dem Resultat zufrieden sind, sperren Sie die Ebene, um ungehindert auf dem Hintergrund arbeiten zu können.

Das waren auch schon die grundlegenden Arbeitsschritte. Ab jetzt geht es ans Eingemachte, der eigentlichen Programmierung des Spiels.

Der Balken

In diesem Schritt werden Sie das erste Grundelement des Spiels, nämlich den Balken, erstellen und ihm ein Skript zuweisen, das die Steuerung per Tastatur ermöglicht.

Der Balken soll vom Benutzer durch die Pfeiltasten gesteuert werden. Außerdem dient er den Bällen als Kollisionsfläche, welche die Bälle in Gegenrichtung abprallen lässt.

Öffnen Sie über Einfügen·Neues Symbol den Symboleigenschaftendialog. Geben Sie dem neuen Symbol den Namen »Balken«, und wählen Sie die Option Filmsequenz. Bestätigen Sie mit OK. Nun befinden Sie sich im Bearbeitungsmodus der Filmsequenz. Legen Sie zwei neue Ebenen an, so dass Sie nun insgesamt drei leere Ebenen zur Verfügung haben. Benennen Sie Ebene 1 »Aktionen«, klicken Sie doppelt auf das erste Schlüsselbild und öffnen Sie das ActionScript-Eingabefenster.

Fügen Sie dort nun folgende Aktion ein:

```
speed = 10;
```

Diese Aktion bewirkt, dass eine Variable namens speed deklariert und dieser der Wert 10 zugewiesen wird. Die Variable gibt – wie der Name schon sagt – die Geschwindigkeitsvorgabe des Balkens an. Wenn Sie diesen Wert erhöhen bzw. herabsetzen, wird der Spieler den Balken schneller bzw. langsamer bewegen können.

Markieren Sie nun **Ebene 2**, und nennen Sie diese »Grafik«. Zeichnen Sie nach bestem künstlerischen Können einen Balken (oder importieren Sie eine Bilddatei, die Sie vorher in einem Grafikprogramm erstellt haben) und platzieren diesen zentriert auf dem Mittelpunktkreuz.

Ebene 3 wird »Kollisionsfläche« benannt. Dort erstellen Sie – wie schon weiter oben beschrieben – eine neue Filmsequenz, die Sie auch »Kollisionsfläche« nennen können. Im Bearbeitungsmodus zeichnen Sie eine Linie, die exakt die Breite des in Ebene 2 befindlichen Balkens haben muss. Das bewerkstelligen Sie am besten, indem Sie provisorisch eine Linie zeichnen, die Filmsequenz dann aus der Bibliothek (Strg+L) auf Höhe des Balkens ziehen, doppelt anklicken und nun, während die Balkengrafik sichtbar ist, die Linie genau der Breite anpassen können. Positionieren Sie die Filmsequenz so, dass sie genau am oberen Ende der Balkengrafik liegt. Da die Ebene »Grafik« über der Ebene »Kollisionsfläche« liegt, ist die Filmsequenz dahinter nicht zu sehen.

Geben Sie nun der Filmsequenz »Kollisionsfläche« einen Instanznamen, indem Sie die Filmsequenz markieren und über das Menü Fenster·Bedienfelder·Instanz im Feld Name die Instanz mit »hit« benennen. Natürlich können Sie auch jeden anderen Instanznamen einsetzen, doch es ist ratsam, kurze, aussagekräftige und prägnante Bezeichnungen zu wählen, damit es später nicht zu Verwechslungen oder Fehlern in ActionScript kommt. Damit ist die halbe Arbeit getan: Nun folgt die Programmierung der Steuerung.

Programmierung der Steuerung

Wechseln Sie über Bearbeiten·Film bearbeiten (Strg+E) in das Bearbeitungsfenster der Hauptzeitleiste zurück. Positionieren Sie nun die Filmsequenz »Balken« auf der Bühne, indem Sie sie aus der Bibliothek (Strg+L) ziehen. Weisen Sie ihr wie weiter oben beschrieben den Instanznamen »bar« zu.

Markieren Sie die Filmsequenz und öffnen über Fenster·Aktionen (Strg+Alt+A) das ActionScript-Bearbeitungsfenster. Hier werden Sie der Filmsequenz selbst nun den Code zuweisen, der nötig ist, um die Steuerung per Tastatur zu realisieren:

```
onClipEvent (enterFrame) {
   if (Key.isDown(Key.LEFT)) {
      this._x -= speed;
   }
   if (Key.isDown(Key.RIGHT)) {
      this._x += speed;
   }
   if (this._x>376) {
```

```
      this._x + = 0;
   }
   if (this._x<0) {
      this._x + = 376;
   }
}
```

Sehen wir uns den Code Zeile für Zeile an:

```
onClipEvent (enterFrame) {
```

Das Ereignis **enterFrame** der Funktion **onClipEvent** tritt ein, sobald die Filmsequenz abgespielt wird und ein Bild sichtbar ist.

Nun wird eine **if**-Konstruktion eingeleitet.

```
if (Key.isDown(Key.LEFT)) {
```

Dem Key-Objekt wurde die Methode **isDown** zugewiesen. Sobald die Taste in () der Methode **isDown** (in diesem Fall **Key.LEFT,** die linke Pfeiltaste) gedrückt ist, wird **true** als Ergebnis der **if**-Abfrage ausgegeben und die folgende(n) Aktion(en) ausgeführt, nämlich

```
this._x -= speed;
```

this bezeichnet als Ziel sich selbst, also die eigene Filmsequenz. Die Eigenschaft **_x,** die X-Positionierung des Objekts nach **this.** bezeichnet die Eigenschaft, die verändert werden soll. Der Operator -= setzt die Eigenschaft **_x** des Objekts **this** auf den Wert von **this._x** minus den Wert der Variablen **speed**, die wir im ersten Bild der Filmsequenz deklariert haben. Alternativ könnte man auch, wie es in Flash 4-Syntax üblich war, explizit

```
this._x = this._x - speed
```

schreiben, doch der Operator -= verkürzt diese Aktion auf **this._x -= speed**, die folglich dasselbe Resultat erzielt.

Sobald also nun die linke Pfeiltaste als gedrückt erkannt wird, setzt sich die Filmsequenz selbst in der X-Positionierung um den Wert **speed** nach links, da der Wert durch den Operatoren -= vom aktuellen X-Wert abgezogen wird.

```
if (Key.isDown(Key.RIGHT)) {
      this._x += speed;
   }
```

Diese Anweisungen stellen das Äquivalent zur ersten `if`-Abfrage dar, denn hier wird der Methode `isDown` statt `Key.LEFT` nun `Key.RIGHT` zugewiesen, und falls diese Pfeiltaste als aktiv erkannt wird, setzt sich die Filmsequenz um den Wert `speed` nach rechts, da diesmal der Operator += zum Einsatz kommt.

```
if (this._x>376) {
   this._x = 0;
}
if (this._x<0) {
   this._x = 376;
}
```

Sollte der Benutzer den Balken über den Bildschirm hinaussteuern, d.h, der X-Positionswert betrüge >376 oder <0, so wird der Balken durch die-Anweisungen `this._x = 0;` bzw. `this._x = 376;` am entsprechend gegenüberliegenden Ende wieder eingeblendet.

Nach exakter Eingabe des Codes sollten Sie nun sofort die Funktionalität testen, indem Sie das ActionScript-Fenster schließen und über das Menü Steuerung·Film testen (Strg+↵) das Ergebnis begutachten.

Sie haben hiermit die Hälfte der Programmierarbeit hinter sich gebracht und können sich nun an die Erstellung der Spielbälle wagen.

Der Ball

In diesem Schritt werden Sie lernen, wie Sie das zweite Hauptelement, den Ball, animieren und die Kollision mit den Wänden sowie dem Balken feststellen und den Ball dementsprechend reagieren lassen.

Kommen wir nun zu den Vorbereitungen:

Wie auch der Balken, so muss auch der Ball als eigene Filmsequenz angelegt werden. Öffnen Sie über Einfügen·Neues Symbol den Symboleigenschaftendialog. Geben Sie dem neuen Symbol den Namen »Ball«, und wählen Sie die Option Filmsequenz. Bestätigen Sie mit OK. Nun befinden Sie sich im Bearbeitungsmodus des Balls. Legen Sie eine neue Ebene an, so dass Sie nun insgesamt zwei leere Ebenen zur Verfügung ha-

ben. Benennen Sie Ebene 1 »Aktionen« und Ebene 2 »Ball«. Hierauf klicken Sie auf das erste Schlüsselbild der Ebene »Aktionen« und öffnen Sie das ActionScript-Eingabefenster.

Fügen Sie folgendes Skript ein:

```
bx_speed = 6;
by_speed = 6;
stop ();
```

Wie schon bei der Erstellung des Balkens wird dem Ball eine Geschwindigkeitsvorgabe gesetzt, doch in diesem Fall nicht durch eine, sondern durch zwei Variablen. Der Grund hierfür ist, dass – im Gegensatz zum Balken – die Bälle nicht nur horizontal bewegt werden, sondern diagonal, also auf der vertikalen und horizontalen Ebene. Die Variable **bx_speed** mit dem Wert 6 beinhaltet die X-Geschwindigkeit (die Horizontale), die Variable **by_speed**, ebenfalls mit dem Wert 6, dagegen die Y-Geschwindigkeit (die Vertikale). Dafür, dass die Filmsequenz im ersten Bild stehen bleibt, da im nächsten Schritt noch ein zweites Bild hinzugefügt wird, sorgt die Aktion `stop ();`.

Schließen Sie nach Eingabe des Codes das Fenster, markieren Sie das immer noch in der Ebene »Aktionen« befindliche zweite Bild, und drücken Sie F7. Damit fügen Sie ein zweites Schlüsselbild ein. Nach einem Doppelklick auf selbiges finden Sie sich im ActionScript-Fenster wieder, in dem Sie folgende Aktionen setzen:

```
_root.balls -= 1;
stop ();
```

Die Aktion `_root.balls -= 1;` bewirkt, dass bei »Betreten« des Bildes in der Hauptzeitleiste (`_root`) die Variable **balls** um den Wert 1 durch den Operator `-=` verringert wird. Die Variable **balls** werden wir im Folgenden noch näher behandeln. Auch in diesem Bild hält eine `stop ();`-Aktion die Filmsequenz in diesem Bild an.

Markieren Sie nun Bild 1 in der Ebene »Ball«. Zeichnen Sie dort mit dem Kreistool einen Ball, positioniert sein muss, so dass in der linken oberen Ecke das Mittelpunktkreuz liegt. Achten Sie darauf, dass der Kreis nur in Bild 1 sichtbar ist. In Bild 2 darf er nicht präsent sein.

Wechseln Sie nun über das Menü BEARBEITEN · FILM BEARBEITEN (Strg+E) zurück auf die Hauptzeitleiste. Ziehen Sie jetzt die eben erstellte Filmsequenz »Ball« aus der Bibliothek (Strg+L) an eine belie-

bige Stelle auf die Bühne. Geben Sie ihr über Fenster·Bedienfelder·Instanz im Feld Name die Bezeichnung »ball«.

Nun kommt der filigrane Teil. Im folgenden Schritt müssen Sie die X- und Y-Koordinaten der Wände des Spielbereichs feststellen. Dazu öffnen Sie über Fenster·Bedienfelder·Info das Informationsfeld. Im ersten Register lassen sich in den Feldern X und Y die Koordinaten des im Moment aktiven Objekts ablesen.

Markieren Sie nun die Filmsequenz »ball«, und positionieren Sie sie so auf dem Hintergrund, als ob der Ball gerade an die linke Wand (Höhe ist nicht relevant) prallen würde. Im Feld X können Sie nun die X-Position des Balls auf der Bühne erkennen. Schreiben Sie sich diesen Wert auf (wir wollen als Beispiel annehmen, der Wert betrüge 59.

Verschieben Sie den Ball nun so, als würde er gegen die rechte Wand prallen (Höhe ist nicht relevant). Merken Sie sich wiederum den Wert, der im X-Feld angezeigt wird (angenommen 302).

Jetzt ziehen Sie den Ball in eine Position, in der er mit dem oberen Teil der Wand kollidieren würde. Notieren Sie sich diesmal nicht den X-, sondern den Wert des Y-Feldes (angenommen 29).

Im letzten Schritt setzen Sie den Ball in den Bereich unter dem Balken und schreiben Sie wiederum den y- Wert auf (angenommen 378).

Mit diesen vier Werten ist es uns nun möglich, zu testen, ob der Ball an die linke oder rechte Wand (X-Werte wie im Beispiel 59 und 302) stößt, in Berührung mit der oberen Wand kommt (erster Y-Wert 29) oder ob er sich über den Balken hinaus nach unten bewegt (zweiter Y-Wert 378).

Programmierung des Balls

Das Skript dazu weisen wir dem Ball zu, indem wir ihn markieren und über das Menü Fenster·Aktionen (Strg+Alt+A) das ActionScript-Bearbeitungsfenster öffnen:

```
onClipEvent (load) {
   this._x = random(221)+80;
   this._y = random(101)+50;
}
onClipEvent (enterFrame) {
   this._x += bx_speed;
   this._y += by_speed;
   if (this.hitTest(_root.bar.hit)) {
```

```
      by_speed = -by_speed;
   }
   if (this._x<59 || this._x>302) {
      bx_speed = -bx_speed;
   }
   if (this._y<29) {
      by_speed = -by_speed;
   }
   if (this._y>378) {
      this.nextFrame();
   }
}
```

Auch diesen Code gehen wir wieder Zeile für Zeile durch:

```
onClipEvent (load) {
```

Das Ereignis `load` der Funktion `onClipEvent` tritt ein, sobald die Filmsequenz in der Hauptzeitleiste geladen wird. Hierauf werden folgende Aktionen ausgeführt:

```
this._x = random(221)+80;
this._y = random(101)+50;
```

`this` bezieht sich auf die eigene Filmsequenz als Ziel. Der Wert der Eigenschaft `_x` (X-Position) wird auf `random (221) + 80` gesetzt. Der Ausdruck `random (221)` erzeugt eine Zufallszahl zwischen 0 und 220. Zu dieser Zufallszahl wird 80 addiert, so dass der mögliche Werteradius zwischen 80 und 300 liegt. Also wird die Filmsequenz auf einen der Werte zwischen 80 und 300 im X-Koordinatensystem gesetzt. Bei der Random-Einstellung ist es wichtig, zu beachten, dass die Zufallszahl nicht im Koordinatenbereich der Hintergrundwände liegt, da sonst der Fall »auf die Wand« gesetzt werden würde. Um dies zu verhindern, vergleichen Sie die möglichen `random`-Werte mit den Koordinatenzahlen, die Sie vorhin notiert haben, als Sie die Kollisionsgrenzen feststellten. Die Ergebnisse dürfen nicht kleiner als der X-Wert der linken und nicht größer als der der rechten Wand sein (konkret also hier nicht kleiner als 59 und nicht größer als 302.

Analog dazu wird durch den Ausdruck `random(101)+50;` ein Zufallswert zwischen 100 und 150 für die Y-Position des Balles ausgegeben (`random (101)` erzeugt eine Zahl zwischen 0 und 100; zu dieser wird 50 addiert).

Diese Zufallspositionierung des Balls wird nur ein Mal ausgeführt, nämlich dann, wenn die Filmsequenz in der Hauptzeitleiste geladen wird.

```
onClipEvent (enterFrame) {
```

Das Ereignis **enterFrame** der Funktion **onClipEvent** tritt ein, sobald die Filmsequenz abgespielt und ein Bild sichtbar wird.

```
this._x += bx_speed;
this._y += by_speed;
```

Die Filmsequenz versetzt sich selbst (**this**) um den Wert der aktuellen X-Position plus **bx_speed.** Analog dazu wird die Position des Balles um den aktuellen Wert der Y-Position plus **by_speed** geändert. Das stellt die Bewegung des Balls dar, denn wiederholt eingesetzt wird der Ball immer wieder um **bx_speed** und **by_speed** verrückt. Die beiden Variablen wurden übrigens im ersten Frame der Filmsequenz definiert, und deren Werte könnten dort zwecks Geschwindigkeitserhöhung bzw. -verringerung variiert werden.

```
if (this.hitTest(_root.bar.hit)) {
```

Die erste IF-Konstruktion wird eingeleitet. Hier kommt die erste Kollisionsabfrage per **hitTest** zum Einsatz. **this** stellt den Ausgangspunkt der Abfrage dar, in **()** ist das Ziel angegeben, nämlich die Filmsequenz mit dem Instanznamen »hit« in der Filmsequenz »bar«, die sich auf der Hauptzeitleiste befindet (**_root**). Diese Instanznamen wurden weiter oben bei der Balkenerstellung vergeben. Im Klartext wird bei Kollision des Balles mit dem Balken der Wert **true** aus der **if**-Abfrage gegeben und die folgende(n) Aktion(en) ausgeführt:

```
by_speed = -by_speed;
```

Der Variablen **by_speed** wird ein negativer Wert zugewiesen. Da sich die Filmsequenz Ball vorher um den Wert **by_speed** auf der y-Achse nach unten bewegt hat, wird durch die »Negativierung« der Variablen die Bewegungsrichtung umgekehrt, und auf der y-Achse bewegt sich der Ball nun nach oben. Damit wird ein Abprallen des Balles auf den Balken im selben Winkel bewirkt, wie der Ball aufgetroffen ist.

```
if (this._x<59 || this._x>302) {
```

Nun wird eine zweite IF-Konstruktion eingeleitet. Jetzt kommen die beiden zuvor ermittelten X-Werte der Wandkollision zum Einsatz. Wenn der **_x**-Wert der Filmsequenz (**this**) kleiner als < 59 oder (||) größer als > 302 ist, wird die folgende Aktion ausgeführt:

```
bx_speed = -bx_speed;
```

Der Variablen **bx_speed** wird ein negativer Wert zugewiesen. Da sich die Filmsequenz Ball vorher um den Wert **bx_speed** auf der x-Achse nach rechts bewegt hat, wird durch die Negativierung der Variablen die Bewegungsrichtung umgekehrt, und auf der x-Achse bewegt sich der Ball nun nach links. Damit wird ein Abprallen des Balls an der linken (respektive rechten) Wand im selben Winkel bewirkt, wie der Ball aufgetroffen ist.

```
if (this._y<29) {
```

Die Aktionen unterhalb dieser IF-Konstruktion werden nur dann ausgeführt, wenn der y-Positionswert des Balls kleiner als < 29 ist. Den Wert 29 haben wir weiter oben als den Wert ermittelt, an dem der Ball mit der oberen Wand kollidiert.

```
by_speed = -by_speed;
```

Wenn die IF-Konstruktion wahr ist, dann wird der Wert der Variablen **by_speed** »negativiert«. Das bewirkt wie schon bei der Kollision mit dem Balken die Umkehrung der Bewegung im Bereich der y-Koordinaten und simuliert so einen Abprall von der oberen Wand. Sollte der Wert schon negativ sein, so wird er durch diese Aktion wieder positiv, denn **- (-by_speed) = by_speed**, da Minus und Minus ein Plus ergeben.

```
if (this._y>378) {
```

Die letzte IF-Konstruktion wird eingeleitet. Sollte der **_y**-Wert des Balls größer als > 378 sein, befindet sich der Ball unterhalb des Balkens (dieser Wert wurde weiter oben ermittelt), die Bedingung ist wahr, und die folgende Aktion wird ausgeführt:

```
this.nextFrame();
```

Die Filmsequenz springt zum nächsten Bild. Noch einmal zur Erinnerung: Im zweiten Bild der Filmsequenz »Ball« befindet sich keine Zeichnung des Balls mehr, und so scheint dieser als »verschwunden«. Außerdem wird in diesem Bild die Variable `balls` um 1 verringert, was also anzeigt, dass ein Ball weniger im Spiel ist.

Mit dieser Aktion endet das Skript. Sie können nun das Resultat betrachten, indem Sie über das Menü STEUERUNG · FILM TESTEN (Strg+↵) in den Filmtestmodus wechseln.

Ebene Aktionen/Einstellungen

Nachdem Sie die Ebenen »Interface«, »Balken« und »Ball« fertig gestellt haben, folgt nun die Ebene »Aktionen/Einstellungen«.

Falls Sie sich nicht im Bearbeitungsmodus der Hauptzeitleiste befinden, wechseln Sie dorthin per BEARBEITEN · FILM BEARBEITEN (Strg+E).

Markieren Sie die Ebene »Aktionen/Einstellungen«, und klicken Sie doppelt auf das erste Schlüsselbild. Im nun geöffneten ActionScript-Fenster wird folgendes Skript zugewiesen:

```
stop ();
balls = 4;
time = 0;
for (i=1; i<balls; i++) {
   duplicateMovieClip ("ball", "ball"+i, i);
}
```

Die Aktion `stop ();` bewirkt, dass der Film an diesem Punkt der Zeitleiste anhält.

```
balls = 4;
```

Die Variable `balls` wird deklariert und der Wert 4 zugewiesen. Diese Variable bestimmt, wie viele Bälle im Spiel vorkommen werden. Durch die letzte Kollisionsabfrage des Ballskripts wird ihr Wert bei Eintreten der Bedingung verringert.

```
time = 0;
```

Der Variablen `time` wird der Wert 0 zugewiesen. Auf diese Variable wird im Folgenden näher eingegangen.

```
for (i=1; i<balls; i++) {
```

Eine FOR-Schleife wird eingeleitet. `i=1` deklariert die Zählervariable i mit dem Wert 1. `i<balls` bildet eine Bedingung. Solange die Variable i kleiner als die Variable `balls` (deren Wert 4 beträgt) ist, werden die folgenden Aktionen ausgeführt. Durch `i++` wird zur Variablen `i` bei jedem neuen Durchgang 1 addiert.

```
duplicateMovieClip ("ball", "ball"+i, i);
```

Die Filmsequenz `ball` wird dupliziert. Der neue Name der Kopie lautet `"ball" + i`, im ersten Durchgang der Schleife also `ball1`, im zweiten `ball2` und im dritten `ball3`, da die Variable `i` nach jedem Durchgang der Schleife durch den Ausdruck `i++` um 1 erhöht wird. Das einzelne `i` bei `duplicateMovieClip` setzt die Tiefe des Duplikats fest, also quasi die »Ebenenschicht«, in die das Duplikat angelegt wird. Da `balls` in der Vorgabe den Wert 4 hat, wird die Aktion drei Mal ausgeführt, also drei Duplikate des Originalballs angelegt, so dass der Spieler insgesamt vier Bälle zu Gesicht bekommt.

Schließen Sie nun das ActionScript-Fenster, und markieren Sie, während die Ebene »Aktionen/Einstellungen« immer noch aktiv ist, das zweite Bild. Durch Drücken von F7 wird ein leeres Schlüsselbild eingefügt, auf das Sie klicken. Im ActionScript-Editor geben Sie nur eine `stop ();`-Aktion ein, und die Ebene ist damit komplett fertig gestellt.

Ebene Controller/Time

Im vorletzten Schritt erstellen wir nun eine Kontrollfunktion, die zum einen die Spielzeit anzeigt und überwacht und zum zweiten ständig abfragt, wie viele Bälle sich noch im Spiel befinden. Sollten keine Bälle mehr im Spiel sein, so wird der bekannte Game Over-Bildschirm angezeigt.

Öffnen Sie über EINFÜGEN·NEUES SYMBOL den Symboleigenschaftendialog. Geben Sie dem neuen Symbol den Namen »Controller«, und wählen Sie die Option FILMSEQUENZ. Bestätigen Sie mit OK. Nun befinden Sie sich im Bearbeitungsmodus der Filmsequenz. Nennen Sie die leere Ebene »Aktionen«, und klicken Sie auf das erste Schlüsselbild. Im ActionScript-Editor tippen Sie folgenden Code ein:

```
_root.time += 1;
if (_root.time >= 60) {
   _root.gotoAndStop(2);
}
if (_root.balls == 0) {
   _root.gotoAndStop(2);
}
```

Im Folgenden die Erläuterungen zu jeder einzelnen Zeile:

```
_root.time += 1;
```

Die Variable `time`, die sich in der Hauptzeitleiste befindet (`_root`), wird um 1 erhöht (`+=`). `time` gibt die abgelaufenen Sekunden seit Spielbeginn aus. Der Grund, warum die Funktion `getTimer ();` nicht verwendet wird, ist, dass diese Funktion nur die Zeit seit Abspielbeginn des Flash-Films misst. Da diese Spielversion nur das Nötigste beinhaltet und Sie höchstwahrscheinlich erst noch einen Startbildschirm mit den Spielregeln etc. davor »schalten« wollen, würde `getTimer ();` die Zeit angeben, die seit Beginn des Flash-Filmabspielens vergangen ist und somit nicht die reale Spielzeit. Deswegen wird die Zeitausgabe wie im Folgenden beschrieben nur »emuliert«.

```
if (_root.time >= 60) {
```

Eine IF-Konstruktion wird eingeleitet, die bedingt, dass die folgenden Aktionen nur dann ausgeführt werden, wenn die Variable `time` der Hauptzeitleiste `(_root)` größer `>` oder gleich `=` 60 ist. Dann ist eine Spielminute vergangen.

```
   _root.gotoAndStop(2);
```

Wird die oben genannte Bedingung erfüllt, so springt der Film (`gotoAndStop`) in der Hauptzeitleiste `_root` zu Bild 2.

```
if (_root.balls == 0) {
```

Eine weitere IF-Konstruktion: Wenn der Wert der Variablen `balls`, die sich in der Hauptzeitleiste `_root` befindet, gleich == 0 ist, dann ...

```
_root.gotoAndStop(2);
```

... springt der Film (**gotoAndStop**) in der Hauptzeitleiste **_root** zu Bild 2. Damit wäre der erste Teil des Controllers »geschrieben«. Damit die Timervariable aber immer um 1 erhöht wird, und dass nicht beliebig und auch noch im Sekundentakt, müssen wir also diese Aktionen wiederholt abspielen lassen. Zu Beginn des Kapitels haben Sie bei den Grundeinstellungen die Bildwiederholrate des Films auf 20 fps (frames per second = Bilder pro Sekunde) gestellt. Wenn Sie nun also 19 Bilder später eine Aktion **gotoAndPlay (1);** einfügen, dann werden die Aktionen von Bild 1 alle 20 Bilder = 1 sek. wiederholt, und somit kann die Zeitanzeige steigen und die Ballanzahlabfrage immer wiederholt werden.

Wechseln Sie nun über das Menü Bearbeiten · Film bearbeiten (Strg+E) zurück auf die Hautpzeitleiste. Markieren Sie das erste Bild der Ebene »Controller/Time«, und ziehen Sie jetzt die eben erstellte Filmsequenz »Controller« aus der Bibliothek (Strg+L) irgendwohin außerhalb des Bühnenbereichs.

Zwar wird nun die Variable **time** im Sekundentakt hochgezählt, nur wird sie dem Spieler nicht angezeigt. Das werden wir nun ändern. Wählen Sie dazu das Menü Fenster · Bedienfelder · Textoptionen. Mit dem Texttool klicken Sie auf den unteren Bereich des Interfaces und wählen dann im Pull-down-Menü von Textoptionen · Dynamischer Text. Tippen Sie im Feld Variable »time« ein, und entfernen Sie das Häkchen bei Auswählbar und Rand/Hintergrund. Jetzt wird der Wert der Variablen **time** während des Spiels in diesem Textfeld angezeigt.

Damit haben Sie die Kontrollfunktion erfolgreich programmiert und können sich nun dem letzten und leichtesten Schritt widmen.

Ebene Text/Buttons

Um Ihrem Spieler die Möglichkeit zu geben, das Spiel zu wiederholen, falls er verloren hat oder das Zeitlimit überschritten wurde, gestalten Sie jetzt noch einen Game Over-Bildschirm.

Markieren Sie dazu in der Ebene »Text/Buttons« das zweite Bild und drücken F7. Dadurch fügen Sie ein leeres Schlüsselbild ein, das Sie nun bearbeiten werden. Schreiben Sie mit dem Textwerkzeug »Game Over« und platzieren die Schrift mittig auf das Interface. Erstellen Sie nun über das Menü Einfügen · Neues Symbol noch einen »Play again?«-Button, indem Sie bei Verhalten nun Schaltfläche selektieren. Im Bearbeitungs-

modus gestalten Sie die einzelnen Buttonphasen nach Ihren Wünschen. Kehren Sie dann per Bearbeiten·Film bearbeiten (Strg+E) zur Hauptzeitleiste zurück. Ziehen Sie die Schaltfläche aus der Bibliothek (Strg+L) an einen Punkt Ihrer Wahl auf die Bühne. Klicken Sie nun mit der rechten Maustaste auf den Button und wählen Aktionen. Im Action-Script-Fenster tragen Sie nun das letzte Skript des Spiels ein:

```
on (release) {
   gotoAndStop (1);
}
```

Diese Aktionen bewirken, dass nach einem Klick auf den Button und anschließendem Loslassen der Film zurück zu Bild 1 springt und somit ein weiteres Spiel ermöglicht. Die Werte der Variablen werden in der Ebene »Aktionen/Einstellungen« im ersten Bild auf die Ursprungswerte zurückgesetzt, und das Spiel kann von neuem beginnen.

So, nun haben Sie sich wohl durchgeboxt und Ihr erstes kleines Spiel programmiert. Wie Sie sehen, sieht die Beschreibung komplizierter aus als die Durchführung im Endeffekt wirklich ist. Um dem Spiel Ihre persönliche Note zu verleihen, können Sie verschiedene Grafiken verwenden. Die Bälle müssen ja nicht unbedingt eine runde Form besitzen, es könnten auch Kokosnüsse oder Ihr Firmenlogo sein.

Ich hoffe, Sie hatten ein wenig Freude, diese Anleitung nachzuvollziehen, und sind jetzt auf den Geschmack gekommen, ein eigenes Spiel zu entwerfen. Viel Spaß und Erfolg dabei ... und bedenken Sie eines: Grenzen setzt Ihnen nur Ihre Fantasie!

Flash-Huhn, Slotmaschine und Co.

Der Spaß des Flashers, sich in komplexen Denkmodellen zu verschachteln, kann nirgends so perfektioniert werden wie in der Spieleprogrammierung. Wir möchten Ihnen anhand einiger Beispiele zeigen, wie Teile von Spiele-Engines gelöst werden können.

Flash-Huhn

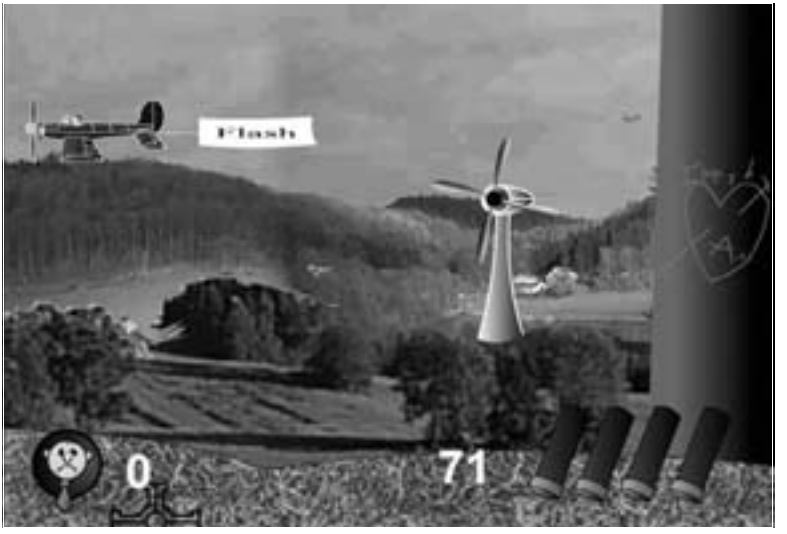
▸ Peter Krempl
www.flashspiele.de

Natürlich juckt es jeden in den Fingern, der ein Flash-Spiel bauen will, auch einmal ein Flash-Huhn zu fertigen. Eigentlich ist es ja ein recht einfaches Spielprinzip. Es wird auf sich bewegende Buttons geklickt, die daraufhin nach unten verschwinden. Mit einer scrollbaren Bühne kann man auf die Pirsch gehen. Der eigentliche Spielspaß kommt durch kleine eingebaute Überraschungen. Die vorliegende Datei auf der CD soll jedoch nur das grundlegende Prinzip des Spieles erläutern.

Erst einmal benötigt man Zielobjekte. Dieses Objekt, nennen wir es das »Flash-Huhn«, muss aus dem Dickicht aufsteigen und langsam über den Screen fliegen. Dazu erhöhen wir seinen `_y`-Wert permanent um den Faktor `stepy`. Damit es nicht in den Äther entschwindet, senken wir den Faktor kontinuierlich, so dass er wieder auf 0 wandert. Das Huhn hält somit seine Höhe.

Um es in der x-Achse zu animieren, geben wir den Wert `stepx` zum `_x`-Wert dazu, den wir, nun andersherum, langsam heraufsetzen. Das Flash-Huhn nimmt in horizontaler Richtung Fahrt auf. Gleich verschwindet es am Bildschirmrand.

Damit wir ihm folgen können, sollten wir wie im Original zur Seite scrollen können. Hier ist dies so gelöst, dass der Hintergrund aus einem Tweening besteht, in dem zwei Hintergrundebenen verschieden schnell seitlich bewegt werden. Um das Tweening in zwei Richtungen laufen lassen zu können, wird es nicht mit `Play` angesprochen, sondern mit `Go to and stop ( _currentframe +` oder `–1`). Dieser Befehl kommt vom gezogenen Fadenkreuz, das je nach seinem eigenen `_x`-Wert das Tweening vor



◀ **Abbildung 1**
Das Flash-Huhn-Spiel

und zurück laufen lässt. Sobald das Tier endgültig auch den breiten scrollbaren Bereich verlassen hat, setzen wir seinen _x-Wert mit Random auf eine Zufallszahl und seinen _y-Wert auf einen hohen Wert, der im unteren Strauchwerk liegen sollte. Alle anderen Flugvariablen werden auf den Ausgangspunkt gesetzt. Das Huhn kann wieder abheben. Dasselbe machen wir, wenn es einen zu hohen _y-Wert besitzt, dies geschieht jedoch nur, wenn es abgeschossen wurde. Dazu später mehr ...

Jetzt haben wir eine Scrollbühne und ein Flash-Huhn, das fliegt. Der Huhn-MovieClip lässt sich manuell in mehreren Instanzen vervielfachen. Ein paar von den Hühnern sollte man umdrehen, um sie in die andere Richtung fliegen zu lassen. Davor haben wir unser Huhn noch verwundbar gemacht, indem es einen (nicht zu kleinen!) unsichtbaren Button erhält, der mit folgenden Set Property-Aktionen gespickt ist:

- `stepx` auf 0 setzen
- `stepy` heraufsetzen
- Drehung auf eine nette Rückenlage
- in den MovieClip integrierten Punktefilm starten (10 bzw. 25 Punkte)
- das Punktekonto in der Hauptzeitachse um eben diesen Punktewert erhöhen
- den Sound-MovieClip in der Hauptzeitleiste starten, um es gackern zu hören.

Sobald das Huhn durch die erhöhte Y-Verschiebung am Boden liegt, erfährt es also, ob es den scrollbaren Bereich verlässt. Das Huhn ersteht nun von den Toten auf und startet zu einem neuen Flug. Hierbei wird jedoch zusätzlich noch die Rückenlage zurückgesetzt, indem man die Drehung wieder auf den Ausgangswert »0« setzt.

Das Schussgeräusch stammt von einem unsichtbaren Button, der die ganze Bühne einnimmt, und vom Treffbutton des Huhnes. Da das Fadenkreuz in einer Ebene über den Hühnern liegt, kann es den Schussbutton nicht beinhalten. Bei jedem Schuss, sei es auf das Huhn oder auf die Fläche, wird ein weiterer MovieClip um ein Bild weitergespielt, der Patronenfilm. Sobald vier Bilder vergangen sind, in denen eine Patrone nach der anderen verschwindet, blendet eine Frame-Aktion einen Abdeck-MovieClip ein, der einen weiteren unsichtbaren, über das ganze Geschehen reichenden Button beinhaltet. Dieser Button verhindert das Schießen und spielt im gedrückten Frame das Geräusch der leeren Büchse ab.

Nachdem man einen Button zum Nachladen (gekoppelt ist die Leertaste) im letzten Frame des Patronenfilms aktiviert, springt der Patronenfilm auf Frame 1 zurück, und der Abdeckfilm verschwindet. Das bedeutet: Feuer wieder frei. Die rechte Maustaste steht in Flash ja leider nicht zur Verfügung.

Nach Lust und Laune kann man nach dem beschriebenen Schema noch kleine Überraschungen einbauen, z. B. den Flieger, den Ballon, die Mühle, den Baum oder was Ihnen noch einfällt. Denken Sie sich etwas Witziges aus.

In der Hauptzeitachse läuft eine Uhr rückwärts, die das Spiel bei 0 beendet. Die Variable `Treffer` zählt die abgeschossenen Hühner. Vor dem Spiel wurde der Name des Spielers abgefragt. Beide Werte, Treffer und Name, werden mit Hilfe eines Tools von Joscha Feth (www.flashtool.de – FlaWRITE.exe, siehe auch Seite 430) in einer Highscore-Liste als TXT-File ausgegeben und im selben Verzeichnis abgespeichert.

Alle drei Dateien, Flash-Huhn.exe, FlaWRITE2.exe und score.txt, müssen im selben Verzeichnis liegen. Da die vorliegende Datei als Offline-Version gedacht ist, die nur mit dem Projektor zu spielen ist, kann man das Spiel mit der Taste »Chef kommt« jederzeit schnell beenden. Danke an Carlo Blatz für die Flash 5-Version.

◀ **Abbildung 2**
Die Minislotmaschine

Minislotmaschine

◀
Peter Krempl
www.flashspiele.de

Flash bietet mit Drag and Drop die schöne Möglichkeit, recht real mit Gegenständen zu hantieren. Warum also beim Wetten nur auf Summen und Buttons klicken und nicht »wirklich« Münzen in die Slotmaschine werfen? Die Datei auf der CD-ROM nennt sich Muenzen.fla.

Mittels einer Berechnung wird im ersten Frame des Films die aktuelle Summe auf die verschiedenen Münzwerte (10, 50, 100) verteilt. Im nächsten Schritt generieren sich aus den verschiedenen Summen, die jede Münzsorte aus der Gesamtsumme zugeteilt bekommen hat, die drei verschiedenen Münzstapel. Dies geschieht durch `duplicateMovieClip` in einer LOOP-Schleife. Die Münzen lassen sich per Drag and Drop in den Einwurf der Slotmaschine werfen. Drücken Sie auf `Play`, um ein Spiel zu simulieren, das entweder einen Gewinn, einen Verlust oder Gleichstand herausgibt. Sofort werden, je nach Gewinn oder Verlust, dem neuen Kassenstand entsprechend die Stapel neu generiert. Der eigene Kassenstand ist nicht nur als Wert wahrnehmbar, er lässt sich auch optisch als Menge erfassen.

Abbildung 3 ▸
Das Flash-Billard
von Peter Krempl

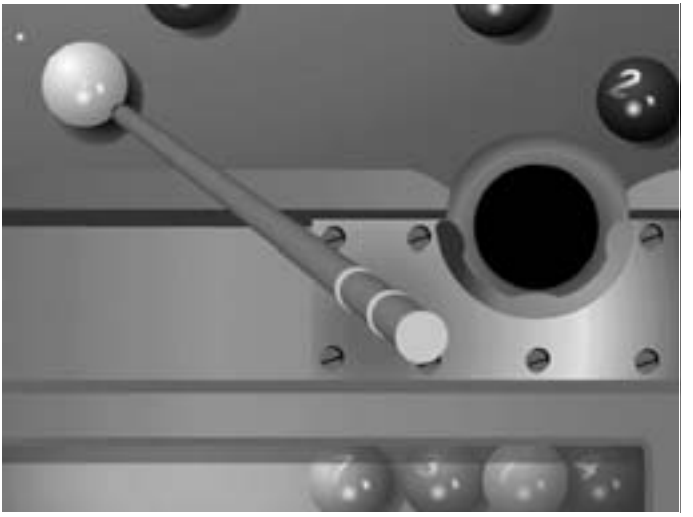
Flash-Billard

▸
Peter Krempl

Das Poolbillard ist komplett mit den begrenzten Möglichkeiten von Flash 4 ActionScript realisiert. Das Grundschema ist folgendes: Die Spielkugel soll durch den Anstoß zwei Geschwindigkeitsvariablen übergeben werden, die stetig reduziert werden müssen (verlangsamen). Dazu wird permanent die Kollision mit Bande und anderen Objekten (Kugeln, Löcher) abgefragt, und bei stattfindender Kollision werden für die beteiligten Kugeln neue Tempovariablen errechnet.

Anstoß und Lauf

Das Anstoßen mit einem virtuellen Queue geschieht auf recht einfache Weise. Der weiße Ball besteht aus einem runden Button, der beim Anklicken die Differenz zwischen den XY-Werten des Mauszeiger und denen der Kugel berechnet. Aus diesen Werten werden die Richtungs- und Geschwindigkeitswerte der Kugel gezogen und dem MovieClip der weißen Kugel übermittelt. Je weiter der Abstand, umso höher die Stoßstärke. Die zwei Geschwindigkeitsvariablen, um die die XY-Werte der Kugel mit Set Property andauernd erhöht/verringert werden, werden selbst zum lang-



◀ **Abbildung 4**
Flash-Billard – Der Duplicate-MovieClip-Queue

samen Abbremsen fortlaufend herabgesetzt, bis sie wieder auf 0 stehen. Die Kugel ruht.

Der Queue ist mit `DuplicateMovieClip` erzeugt und hat dadurch eine Art 3D-Wirkung. Eine Punktlinie zeigt den zu erwartenden Kugellauf an. Der Abstand der Punkte zueinander deutet die Stoßstärke an.

Kollision

Siehe auch auf der CD-ROM Spiele\Flashbillard Abstand_R.fla und Abstand_Q.fla

In Flash 5 steht nun auch die Sinusfunktion zur Verfügung, und Objektkollisionen kann man direkt abfragen. Die Sinusfunktion, die zur Winkelberechnung notwendig ist, lässt sich in Flash 4 nur durch das manuelle Eingeben einer Sinus-Tabelle aus der guten alten Mathe-Formelsammlung (nie alte Schulbücher wegwerfen!) ersetzen. Die Kollisionen lassen sich durch das Abfragen des Abstandes zweier Objekte erkennen. »Runde« Kollisionen benötigen dazu den Satz des Pythagoras. Für »quadratische« Kollisionen genügen allein die Differenzen zwischen den jeweiligen X- und Y-Koordinaten.

Um den Satz des Pythagoras ($a^2 + b^2 = c^2$) in Flash zu berechnen, muss man eine kurze LOOP-Schleife zum Wurzelziehen bemühen. In der Regel genügen jedoch »quadratische« Kollisionen, auch wenn die Objekte bei

45/135/225/315°-Kollisionen einen größeren Abstand haben als bei 0/90/180/270°-Kollisionen. Der Rechenaufwand ist etwas geringer als beim »runden« Zusammenstoß. Hier wurde die einfachere Kollisionsabfrage verwendet. Bei neun Kugeln und sechs Löchern wäre eine permanente Loop-Abfrage zwischen allen beteiligten 15 Objekten einfach zu rechenintensiv gewesen. Die Bandenkollision wird durch eine einfache Umkehrung der jeweiligen XY-Tempovariablen erreicht.

Verhalten

Die Reaktionen der Kugeln auf Kollisionen sind auf die Wirkung ausgelegt. Das heißt, die Kugeln sollen sich annähernd korrekt abstoßen. Es muss echt aussehen, nicht den Newton´schen Lehrsätzen entsprechen! Also wurde das Kollisionsverhalten nur mathematisch angenähert. Soll es den Physikern überlassen sein, das Verhalten von Kollisionen konkret zu beschreiben, nicht den Flashern. Die neuen Geschwindigkeiten der Kugeln addieren/subtrahieren sich aus den zwei Geschwindigkeiten der beteiligten Kugeln, je nach Eigengeschwindigkeit in unterschiedlicher prozentualer Gewichtung. Das Aufeinandertreffen der Kugeln muss eine schnelle Kugel anders ablenken als eine langsame oder ruhende Kugel. Die eigene Geschwindigkeit geht auch in einem höheren Maße in das neue Tempo ein als die Geschwindigkeitswerte der beteiligten Kollisionskugeln.

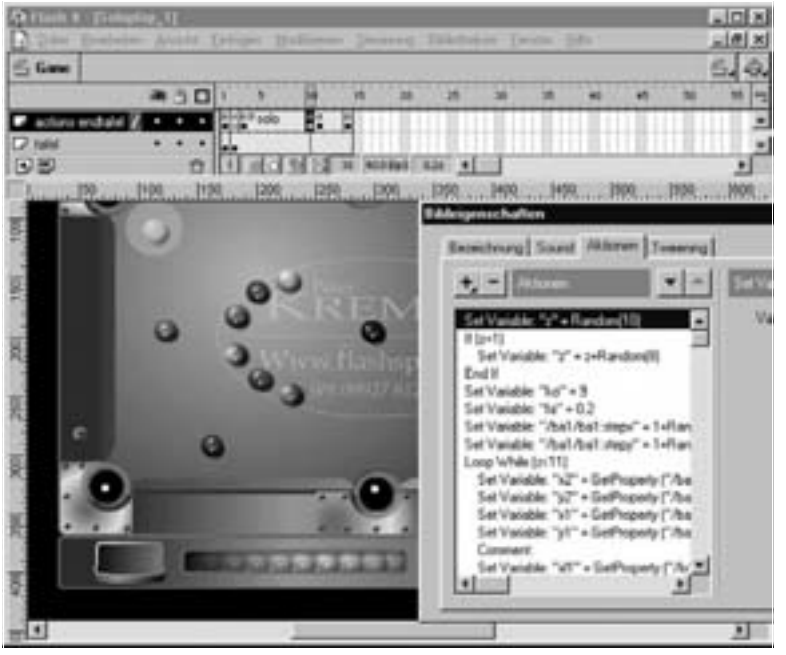
Das Einlochen ist im Prinzip die Kollision mit dem Loch. Es folgen durch ActionScript animierte Bewegungen (Verschwinden der Kugel) und ein normales Tweening beim Abrollen der Kugeln.

Um die Farbreihenfolge beim Abrollen zu wahren, rollen nur verschiedene Instanzen ins Fach. Die Kugeln an sich bestehen aus einem MovieClip, der in neun Keyframes alle verschiedenen Farben beinhaltet. So kann dem MovieClip mitgeteilt werden, aus welcher aktuellen Kugelfarbe er gerade beim Abrollen zu bestehen hat.

Eine Uhr misst die Zeit, die man zum Versenken aller Kugeln benötigt. Für ältere Semester existiert auch eine beschaulichere Karambolage-Version mit drei Kugeln.

Alle Grafiken wurden mit Flash-Bordmitteln erzeugt. Gerade bei rechnerintensiven animierten Spielen sollte man auf aufwändige Pixelgrafiken und aufwändig veränderte Alphawerte verzichten.

Das Loading besteht bereits aus einem Ball, der sich auf einer unsichtbaren Fläche anstoßen lässt. Man kann ihn auf der Fläche so lange wild



◂ Abbildung 5
Flash-Billard – Farbreihenfolge beim Abrollen

herumtreiben, bis das Spiel geladen ist. Gerade Spiele sollten in der Loading-Phase dem Besucher etwas Kurzweil anbieten.

Anmerkung: Leider ist das 9-Ball-Poolbillard mit dem momentanen Flash 5 Plug-in, das viele Bugs enthält, nicht zu spielen, da es die Geschwindigkeit erheblich beeinträchtigt hat. Das 6-Ball-Poolbillard und Karambolage sind jedoch auch mit dem Flash 5-Plug-in zu spielen, bitte besuchen Sie hierzu www.flashspiele.de.

Kollisionen, Highscores u.a.

In diesem Kapitel stellen wir Ihnen einige elementare Techniken vor, die zur Programmierung von Spielen nötig sind.

Kollisionen

▶ Peter Krempl www.flashspiele.de

Zur absoluten Kollisionsabfrage sind in Flash 4 Abstandsberechnungen notwendig. Dazu gibt es z. B. zwei verschiedene Möglichkeiten. Einmal die, den absoluten Abstand zweier Objekte mit dem Satz des Pythagoras radialsymmetrisch zu berechnen – und dann noch die einfachere Methode, die Differenzen der jeweiligen X- und Y-Werte in einem quadratischen Koordinatensystem abzufragen.

In Flash5 lassen sich die Kollisionen zweier Objekte mit der Aktion `hitTest` leider nur in der »quadratischen« Form abfragen.

Die Dateien auf der CD-ROM nennen sich: Abstand_Q.fla und Abstand_R.fla im Ordner Spiele\Kollision.

Radial

Auch hier benötigt man erst einmal die Differenzwerte der X- und Y-Koordinaten beider Objekte `/drag` und `/fest`.

```
//Die Koordinaten der zwei Objekte//
Set Variable: "x_drag" = GetProperty ("/drag",_x)
Set Variable: "y_drag" = GetProperty ("/drag",_y)
Set Variable: "x_fest" = GetProperty ("/fest",_x)
Set Variable: "y_fest" = GetProperty ("/fest",_y)
//Daraus die Differenzen Delta x und Delta y//
Set Variable: "del_x" = x_drag-x_fest
Set Variable: "del_y" = y_drag-y_fest
```

Aus diesen Werten ergibt sich ein Vektordreieck mit einem rechten Winkel. Somit kann man die Länge der Hypotenuse mit Hilfe des Satzes des Pythagoras ($a^2 + b^2 = c^2$) errechnen. c ist die Wurzel aus $a^2 + b^2$. Mit folgendem ActionScript errechnet man diesen Wert:

```
Set Variable: "hyp_q" = del_x*del_x + del_y*del_y
Set Variable: "hyp" = 1
Loop While (hyp*hyp<hyp_q)
   // Loop While (c*c<c²) //
   Set Variable: "hyp" = hyp+1
End Loop
Set Variable: "abstand" = hyp
```

Das Wurzelziehen kann man in Flash 4 mit einer LOOP-Schleife erledigen, indem man die angenommene Länge der Hypotenuse in der Schleife um den Faktor 1 (= 1 Pixel) ansteigen lässt, bis der gewünschte ganzzahlige Wert erreicht ist. Man kann den Wert auch schneller steigen lassen, um weniger Rechenzeit zu beanspruchen. Damit wird das Ergebnis aber auch ungenauer.

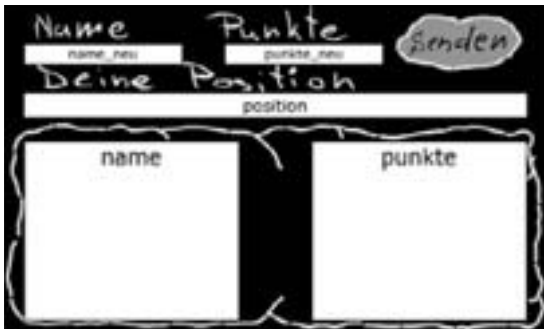
Mit diesem Wert lässt sich jederzeit die Kollision abfragen.

```
If (abstand<(GetProperty ("/drag",_width) + GetProperty ("/
fest",_width) )/2)
   // Als Reaktion wird der MC /fest verschoben //
   Set Property ("/fest", X-Position) = GetProperty ("/fest",_x)-
del_x
   Set Property ("/fest", Y-Position) = GetProperty ("/fest",_y)-
del_y
End If
```

Quadratisch

Hier nimmt man nur die Differenzwerte der Koordinaten und bestimmt die Kollision mit folgendem ActionScript.

```
Set Variable: "kont" = (GetProperty ("/drag",_width) + GetProperty
("/fest",_width))/2
If (del_x<kont and  del_x >-kont and del_y<kont and  del_y >-kont )
   // Als Reaktion wird der MC /fest verschoben //
   Set Property ("/fest", X-Position) = GetProperty ("/fest",_x)-
del_x
   Set Property ("/fest", Y-Position) = GetProperty ("/fest",_y)-
del_y
End If
```



▲ Abbildung 1
Die gestaltete Highscore-Liste

Highscore-Liste in Flash mit PHP4-Skript

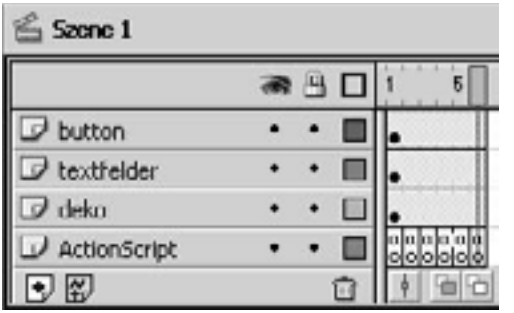
▸ boblgum
www.mysterion.de

Mit diesem Beispiel wollen wir zeigen, dass man bei kleineren Datenmengen auch ohne Datenbankprogrammierung auskommen kann. Des Weiteren werden Sie sehen, wie man Flash mit anderen Skriptsprachen verbinden kann und wie einfach dies in Wirklichkeit ist.

Wir realisieren in Flash 5 eine Highscore-Liste in Verbindung mit PHP. Wir wollten die ganze Sache so einfach wie nur möglich halten und haben deswegen auf die Benutzung einer Datenbank (z.B MySQL) zur Verwaltung der Daten zu Gunsten einer einfachen Textdatei verzichtet. Dabei werden nur grundlegende und unbedingt benötigte Bestandteile des PHP-Skripts näher erläutert. Wer also wissen möchte, wie unser PHP-Skript im Einzelnen aufgebaut ist, müsste zu einem PHP-Buch greifen.

- Aufgabenstellung: In Flash soll eine Highscore-Liste eingelesen, angezeigt und aktualisiert werden. Die aktuellen Daten werden in Flash eingegeben, an PHP-Skript übergeben, dort ausgewertet und von Flash wieder eingelesen und ausgegeben.
- Aufbau: Was wir brauchen ist also ein Flash-Film, in dem wir die Daten eingeben und anzeigen, dann eine Textdatei, in der unsere Daten gespeichert werden, und ein PHP-Skript, das die Daten sortiert und in die Textdatei hinein schreibt.

▲ Abbildung 2
Einstellungen in den Textoptionen



▲ Abbildung 3
Die Keyframes

Aufbau des Flash-Films

Für unser Beispiel benötigen wir fünf Textfelder, in denen verschiedene Variablen ein- oder ausgegeben werden. Die von uns benutzten Variablennamen sind dem Bild 1 zu entnehmen.

Da dies nur eine Simulation einer Highscore-Liste sein soll, müssen wir »name_neu« und »punkte_neu« selbst eingeben. Diese Textfelder müssen deswegen als »Texteingabe« freigegeben werden (siehe Abbildung 2).

Die anderen Textfelder dienen nur zur Ausgabe und werden deswegen unter TEXTOPTIONEN als DYNAMISCHER TEXT eingestellt. Bei den Textfeldern zur Darstellung der Namen und dazugehörigen Punkte muss man auch die Eigenschaft »Eine Zeile« in »Mehrere Zeilen«, aber ohne Wortumbruch, ändern. Zu guter Letzt setzen wir auch noch einen Button ein. Mit dem Klick auf den Button werden die neuen Daten an unser PHP-Skript gesendet, damit die Ausgabe aktualisiert wird.

ActionScript des Films

Wenden wir uns nun dem ActionScript des Films zu. Insgesamt brauchen wir sechs Keyframes (siehe Abbildung 3).

Das Skript im jeweiligen Keyframe der Ebene »ActionScript« sieht folgendermaßen aus:

```
loadVariablesNum ("highscore.txt", 0);
```

Aktionen für Bild 2

```
if (datei_geladen != 1) {
   gotoAndPlay (1);
} else {
   gotoAndStop (3);
}
```

Aktionen für Bild 3

```
stop ();
```

Aktionen für Bild 4

```
loadVariablesNum ("http://localhost/highscore.php", 0, "POST");
```

Aktionen für Bild 5

```
if (geladen == 0) {
   gotoAndPlay (_currentframe-1);
} else {
   gotoAndStop (4);
}
```

Aktionen für Bild 6

```
stop ();
```

Wie man (hoffentlich) unschwer erkennen kann, wird im ersten Keyframe die »alte« Highscore-Liste in den Film geladen. Wir haben außerdem einen kleinen Preloader für die Textdatei eingebaut, der wie folgt funktioniert:

Solange die Variable `datei_geladen` ungleich `1` ist, soll der Film zum Keyframe 1 zurückgehen. Hier ist anzumerken, dass, wenn die Datei am Anfang leer ist, diese Schleife so lange ausgeführt wird, bis die ersten Daten an das PHP-Script gesendet werden, wenn man auf den Button klickt:

```
on (press) {
   gotoAndPlay (4);
}
```

Wir haben hier versucht, auf die Bildbezeichnungen zu verzichten, damit man die Scripterklärung anhand der Bilder nachvollziehen kann.

Wenn man nun auf den Button geklickt hat, springt der Film zum Keyframe 4 und lädt unser PHP-Skript. Dabei werden alle (sehr wichtig) Variablen, die im Film erzeugt wurden, an dieses Skript übergeben. Wie man sieht, wird dieses Skript so lange geladen, bis die Variable `geladen` gleich 1 ist. Woher kommt nun diese Variable? – Aus dem PHP-Skript, wie wir gleich sehen werden.

PHP-Skript (Versuch einer Erklärung)

Als Erstes wird die Variable `geladen` erzeugt, ins Flash-Format umgewandelt, gleich 0 gesetzt und an Flash übergeben:

```
<?
$geladen="&geladen=0";
echo "$geladen";
...
```

Danach wird die Textdatei in die Variable `$datei` eingelesen (was natürlich voraussetzt, daß die Textdatei überhaupt existiert!). Aus der Variablen `$datei` werden zwei Zeilen extrahiert, die in zwei weitere Variablen `$name` und `$punkte` geschrieben werden:

```
...
//vorbereitung der strings "name" und "punkte"
$nl=chr(13).chr(10);

$pos=strpos($datei,$nl);
$name=substr($datei,0,$pos);
$punkte=substr($datei,$pos+2);

$pos=strpos($name,"=");
$name=substr($name,$pos+1);

$pos=strpos($punkte,"=");
$punkte=substr($punkte,$pos+1);
$pos=strpos($punkte,$nl);
$punkte=substr($punkte,0,$pos);
...
```

Diese zwei Variablen werden in einen Array **$score_liste** geschrieben:

```
...
//bilden des arrays
$i=0;

$laenge=strlen($name);
while ($laenge!=0) {
   $z=0;
   $pos=strpos($name,"%0d");
   $name_buffer=substr($name,0,$pos);
   $name=substr($name,$pos+3);

   $pos=strpos($punkte,"%0d");
   $punkte_buffer=substr($punkte,0,$pos);
   $punkte=substr($punkte,$pos+3);

   $score_liste[$i][$z]=intval($punkte_buffer);
   $z++;
   $score_liste[$i][$z]=$name_buffer;
   $i++;

   $laenge=strlen($name);
}
...
```

Die aus Flash übergebenen Werte werden anschließend zum Array hinzugefügt:

```
...
//einfügen der neuen werte in array, die aus flash kommen (name_neu
und punkte_neu)
$score_liste[$i][0]=intval($punkte_neu);
$score_liste[$i][1]=$name_neu;
...
```

Der so erzeugte Array wird dann mit einer Funktion sortiert:

```
...
//sortieren des arrays
   function cmp($score_liste, $b) {
      if ($score_liste[0] == $b[0]) return 0;
      return ($score_liste[0] < $b[0])?1:-1;
   }

usort ($score_liste, cmp);
...
```

Danach werden die einzelnen Zeilen für die Textdatei vorbereitet, ins Flash-Format umgewandelt, und die alte Textdatei wird überschrieben:

```
...
$i=0;
$position=0;

while(list($k, $v) = each($score_liste)) {
   $liste_name.="$v[1]%0d";
   $liste_punkte.="$v[0]%0d";
   $i++;
   if ($punkte_neu == $v[0] && $name_neu == $v[1]) {
      $position = $i;
   }

   if ($i >= 5) break;
}

//textdatei mit neuen werten überschreiben
fputs($fp, "&name=$liste_name$nl");
fputs($fp,"&punkte=$liste_punkte$nl");
fputs($fp,"&datei_geladen=1");

//ermitteln der erreichten position in der highscore-liste
if ($position == 0) {
   $position = "Du bist nicht unter den 5 besten";
} else {
   $position = "Du bist auf dem Platz ".$position;
}
...
```

Die aktualisierten Variablen (**$position**, **$name** und **$punkte**) werden an Flash übergeben:

```
...
//übertragung in flash
$position="&position=".$position;
echo "$position";
$name="&name=".$liste_name;
$punkte="&punkte=".$liste_punkte;
echo "$name$punkte";
...
```

Zu guter Letzt wird die Variable **geladen** gleich 1 gesetzt, damit Flash die Schleife endlich verlassen kann:

```
...
$geladen="&geladen=1";
echo "$geladen";
?>
```

Was bei dieser Vorgehensweise vielleicht zu beachten wäre:

- Es kann passieren, dass mehrere User gleichzeitig ihren Punktestand abspeichern wollen. Was dann passiert, ist noch nicht ausgetestet worden. Wenn man aber bedenkt, wie schnell das Skript abgearbeitet wird, ist die Wahrscheinlichkeit für so eine Situation ziemlich niedrig.
- Außerdem haben wir unsere Highscore-Liste auf die Anzeige der fünf besten eingeschränkt (`if ($i >= 5) break;`). Wenn man also mehr Datensätze in der Textdatei speichern möchte, muss nur die **break**-Bedingung geändert werden: `if ($i >= 50) break;` oder so.
- Des Weiteren sollte die Textdatei möglichst nicht leer sein, wenn man zum allerersten Mal das Skript startet. Der Aufbau der Textdatei könnte so aussehen:

  ```
  &name=test%0dø
  &punkte=0001%0dø
  &datei_geladen=1
  ```

- Alle Dateien, die in diesem Tutorial erwähnt wurden (Flash-Film, PHP-Skript und die Textdatei) müssen im gleichen Verzeichnis auf dem Server, der logischerweise PHP unterstützen muss, abgelegt werden.

Wenn jemand noch nicht wissen sollte, was wir die ganze Zeit mit »Flash-Format« gemeint haben: Damit Flash irgendwelche Variablen aus irgendwelchen Dateien (PHP, ASP, CGI usw.) überhaupt verstehen und verwenden kann, müssen diese folgendes Format haben:

```
&namedervariablen=wertdervariablen
```

Um einen Zeilenumbruch in Flash zu erzwingen, setzen wir »%0d« ein.

Stoppuhr

◀ Peter Krempl

In fast jedem Spiel muss die vergangene Zeit gemessen werden. Mit diesem Beispiel können Sie sich die vergangenen Zehntelsekunden Ihrer Systemuhr in Sekunden, Minuten und Stunden umrechnen und diese analog und digital ausgeben lassen. Die Datei auf der CD-ROM nennt sich zeit.fla.

Um die vergangene Zeit zu ermitteln, bestimmt man die vergangenen vollen 1/10 Sekunden mit `GetTime`. Diesen Wert setzt man alle vollen 10/10 Sekunden auf Null zurück, indem man die zuvor bereits vergangenen Zehntel abzieht (`zeh_alt`)

```
Set Variable: "zeh" = Int(gettime/100)-zeh_alt
```

Somit hat man einen Zähler erzeugt, dessen Wert ein Mal pro Sekunde bis 10 zählt.

Nun muss man immer, wenn eine Sekunde, also zehn Zehntel, verstrichen sind, den Sekundenwert um eins erhöhen. Ebenso berechnet man die vergangenen Minuten und Stunden. Sobald ein Wert unter 10 liegt, wird er für die digitale Ausgabe mit »0«&wert in die zweistellige Form gebracht.

Die analoge Uhr ist denkbar einfach, sind doch Winkel und Zeit recht nah miteinander verwandt. Für jede Sekunde und jede Minute muss der jeweilige Zeiger um 360°/60 sek., min = 6 Grad gedreht werden. Die Zeiger-MCs haben natürlich ihren Mittelpunkt am unteren Ende.

Da der Wert, der den `GetTime` ausgibt, nicht auf null zu setzen ist, muss man, um den Wert null zu erhalten, die vergangene Zeit von sich selbst subtrahieren (= Wert 0) und nur die neuen Differenzen zu diesem Wert ganzzahlig werten.

▲ Abbildung 4
Eine analoge Stoppuhr

Coco-Catch

Ein komplettes Beispiel

Da die Spieleprogrammierung eines der wichtigsten Einsatzgebiete von Flash ist, stellen wir Ihnen hier ein komplexes Spiel komplett vor. Die hier vorgestellte Methode ist nicht die einzige Möglichkeit, dieses Spiel zu programmieren, es geht hier hauptsächlich darum, Ihnen das Arbeiten mit selbst definierten Objekten zu zeigen und Sie zu motivieren, objektorientiert zu programmieren.

▸ Hassan Beigi
www.powerflasher.de

▸ Im folgenden Kapitel möchten wir Ihnen einmal zeigen, wie komplex ein noch so einfach anmutendes Spiel sein kann. Mitte 2000 hat die Traveltainment AG (www.traveltainment.de) die Power und Radach Werbeagentur GmbH (www.power-radach.de) beauftragt, ein Spiel zu kreieren. Es sollte zum Produkt des Kunden passen. Zusammen mit Powerflasher ist ein Jump´n Run-Spiel entstanden, das die Grenzen von Flash ausreizt. Unter www.powerflasher.de/tt kann man es spielen.

Den Original-Sourcecode des Spiels werden wir nicht veröffentlichen, da niemandem daran gelegen ist, dass das gleiche Spiel mit anderen Grafiken im Netz verbreitet wird. Anhand dieses Spiels kann die Komplexität eines solchen Projekts sowie einige Techniken und Denkweisen gut verdeutlicht werden: Die kann auch in anderen Projekten von Nutzen sein.

Daher haben wir das in Flash 4 entwickelte Spiel für dieses Kapitel in Flash 5 ganz neu programmiert. Dank der objektorientierten Programmierung konnten wir es so beschleunigen und übersichtlicher und kleiner programmieren. Die Funktionen möchten wir Ihnen nun erklären:

Ich habe folgende Konventionen für dieses Spiel definiert:

- Alle Koordinatenvariablen haben am Namensende ein X oder Y.
- Alle Variablen, die als Sperre dienen, haben einen Avail als Wortende.

Diese Availvariablen nehmen die booleschen Werte `true` oder `false` an, je nachdem, ob eine Aktion oder Spielfigur gesperrt ist oder nicht. Alle Zufallswerte beginnen mit einem R oder Random als Wortanfang. Alle

◄ **Abbildung 1**
Wir spielen Coco Catch

selbst definierten Objekte fangen mit Obj als Wortanfang an. Alle Hitfelder beginnen mit hitarea als Wortanfang.

Wir werden reine Flash 5-Punktsyntax benutzen.

Überblick über Objekte und Aktionen im Spiel

Grundsätzliches zum Spiel:

- In dem Spiel befinden sich Palmen und Affen, außerdem sind einige Fahrzeuge auf dem Spielfeld positioniert. Diese werfen die Nüsse, die der Player fangen muss.
- Man soll mit der Maus den Player steuern und die Nüsse fangen (der Player wird in dem Code »Knilch« genannt). Bewegt man die Maus, folgt der Knilch der Maus. Mit der linken Maustaste springt er in die Höhe, und wenn er sich an der Bar befindet, gibt er die Nüsse in seinem Netz ab anstatt zu springen.

- Im Spiel soll das Spielfeld gescrollt werden können. Wenn der Knilch am Rand des Spielfeldes ankommt, wird das Spielfeld mit allen Objekten darauf gescrollt.
- Man muss über diverse Hindernisse springen bzw. vermeiden, getroffen zu werden.

Objekte im Spiel und deren Funktion:

- `ObjScroll`: Mit diesem Objekt implementieren wir die Scrollfunktionen in dem Spiel.
- `ObjKnilch`: Mit diesem Objekt wird der Knilch im Spiel gesteuert.
- `ObjKrebs`, `ObjLoch`, `ObjSpinne`: Diese drei Objekte sind für die Hindernisse im Spiel verantwortlich.
- `ObjNuss`: Mit diesem Objekt implementieren wir die Fangfunktion.
- `ObjAffeGen` und `ObjMobileGen`: Diese beiden Objekte sind dafür verantwortlich, die Affen und die Fahrzeuge zu starten.
- `ObjTime`,`ObjScore`: Mit diesen beiden Objekten implementieren wir die Spielzeit- und Score-Funktionen.

Reihenfolge der Implementierung des Spiels

- Zunächst müssen wir alle Grafiken und Animationen der Spielfiguren bereitstellen.
- Optimierung der Animationen, um die Dateigröße so klein wie möglich zu halten. Ziel sind 500 KB mit Sound.
- Erstellung der benötigten Spielfiguren-MovieClips.
- Implementierung der Objekte (der Code zur Steuerung des Spiels).
- Erstellung der Controll-MovieClips zum Aufruf der Methoden der Objekte.

Bei der Bereitstellung der **Grafiken und Animationen** der Spielfiguren mache ich es ganz kurz, denn wie man Grafiken erstellt, soll hier nicht gezeigt werden. Alle Grafiken und Spielfiguranimationen wurden mit FreeHand erstellt. Die drei Hintergrundgrafiken wurden als GIFs mit Transparenz in Flash importiert und verlustfrei veröffentlicht. Die Animationen wurden in FreeHand als SWF exportiert und in Flash importiert.

Zur **Optimierung der Animation** gehen Sie folgendermaßen vor: Die importierten Animationen erscheinen in Flash als Bild-für-Bild-Animation. Als Beispiel nehmen wir die Laufanimation des Knilchs. Der Knilch besteht unter anderem aus folgenden Körperteilen: Kopf, Augen, Oberkörper, Unterkörper, rechter und linker Arm, Hände, Beine, Füße und ein

Netz. Wir nehmen das erste Bild der Animation und machen aus den einzelnen Körperteilen Grafiksymbole, um diese in den folgenden Frames zu benutzen. Statt in jedem Frame jedes Körperteil aufs Neue zu zeichnen, benutzen wir Instanzen dieser Körperteile. Wir positionieren in jedem Frame die Körperteile neu und nehmen als Vorlage für die Positionierung die importierten Bilder aus der FreeHand-Animation. (Hier ist viel Geduld gefragt.)

Diese Schritte führen wir für alle anderen importierten Animationssequenzen durch. Zum Vergleich: Die importierte FreeHand-Laufanimation war 100 KB groß, nach der Optimierung nur noch 13 KB. Das kommt daher, da in der Bild-für-Bild-Animation in jedem Frame jedes Körperteil neu gezeichnet wurde.

Nach einiger Zeit sollten wir nun alle Animationen zur Hand haben:

Der Knilch kann nun in Wartestellung gehen, rennen, springen, stolpern und ins Loch fallen, sich drehen und Nüsse zur Abgabe hochwerfen. Der Krebs kann mit seinen Augen aus dem Sand schauen und nach dem Knilch schnappen. Die Spinne pulsiert, wenn der Knilch in ihre Nähe kommt, und seilt sich ständig auf und ab. Die Affen können jetzt Nüsse werfen. Die Fahrzeuge fahren hin und her und werfen goldene Nüsse.

Nun kommen wir zur Erstellung der benötigten MovieClips.

Wir müssen die verschiedenen Animationsphasen der Spielfiguren in MovieClips positionieren und die drei Spielebenen erstellen.

Die drei Ebenen des Spiels müssen bei Bedarf gescrollt werden. Diese drei Ebenen bestehen zu Beginn aus Hintergrundgrafiken.

- Die dritte Ebene beinhaltet lediglich eine Grafik. Sie stellt die hinterste Ebene im Spiel dar.
- Die zweite Ebene beinhaltet ebenfalls nur eine Grafik. Sie stellt die mittlere Ebene im Spiel dar.
- Die erste Ebene ist die für uns wichtigste. Diese Ebene beinhaltet die Palmen, den Strand und die Bar als eine einzige Grafik.

Wir platzieren zunächst alle drei Ebenen auf der Bühne in der Hauptzeitleiste.

Die erste Ebene erhält den Instanznamen L1, die zweite L2 und die dritte L3, wobei L1 die oberste Ebene der Zeitleiste belegt und L3 die unterste.

In die erste Ebene werden alle Hindernisse, Fahrzeuge, Affen, Nüsse und Fallen platziert, lediglich der Knilch wird in der Hauptzeitleiste platziert.

Sobald der Knilch die Ränder des Spielfeldes erreicht, werden diese drei Ebenen mit unterschiedlichen Geschwindigkeiten gescrollt. Somit erhalten wir den Eindruck einer Pseudo-3D-Umgebung.

Wenn gescrollt wird, müssen alle Objekte auch gescrollt werden, damit die relative Position der Objekte zum Hintergrund nicht verändert wird. Daher müssen alle Objekte in der ersten Ebene L1 platziert werden. Würde man die Objekte nicht in diese Ebene platzieren, würden sich z. B. die Palmen unter den Affen hinweg bewegen.

Um scrollen zu können, platzieren wir zwei unsichtbare Rechtecke an jeder Seite des Spielfeldes. Diese benutzen wir als Hitfelder. Dem linken Hitfeld geben wir den Instanznamen `hitareaLeft`, dem rechten entsprechend den Instanznamen `hitareaRight`. Mit Hilfe dieser zwei Hitfelder können wir später im Code feststellen, ob der Knilch am Rand des Spielfeldes angekommen ist und ob gescrollt werden muss.

Erstellung des Knilchs

Jede Animationsphase des Knilchs kommt in einen eigenen MovieClip.

Wir erstellen sieben MovieClips:

1. Knilch-Wait: Warteanimation
2. Knilch-Run: Rennanimation
3. Knilch-Jump: Sprunganimation
4. Knilch-Deliver: Nussabgabeanimation
5. Knilch-Turn: Drehanimation
6. Knilch-Stumble: Stolperanimation
7. Knilch-Fall: Fallanimation

Der Knilch trägt ein **Netz** mit sich. Dieses Netz dient dazu, die Nüsse zu fangen. Das Netz ist ein MovieClip mit vier Frames. Im vierten Frame sieht man drei Nüsse im Korb, im dritten Frame zwei Nüsse, im zweiten Frame eine Nuss, und im ersten Frame ist nur das Netz zu sehen. Jedes Frame besitzt eine Aktion `this.stop();`. Dieser Netz-MovieClip spielt eine zentrale Rolle im Spiel. Das Netz muss in jeder Animationsphase zu sehen sein.

Für die Kollisionsabfragen werden wir rechteckige Felder positionieren, die als Hitfelder benutzt werden. Die Kollisionsabfragen werden dann mit Hilfe dieser Hitfelder implementiert.

Wir positionieren ein Rechteck über den Fangbereich des Netzes und geben ihm den Instanznamen `hitarea1`. Wenn eine Nuss dieses Rechteck trifft, soll natürlich auch im Netz eine weitere Nuss sichtbar werden, und

die Punkte im Netz müssen sich erhöhen. Dies prüfen wir mit `hitTest`. Wie man eine weitere Nuss im Netz sichtbar machen kann, dürfte dem aufmerksamen Leser nicht entgangen sein: Mit `this.gotoAndStop (3)` sieht man z. B. drei Nüsse im Netz.

Wir positionieren ein weiteres Rechteck direkt hinter dem ersten. Dieses Rechteck erstreckt sich bis knapp hinter den Knilch. Wir geben ihm den Instanznamen `hitarea2`. Wenn eine Nuss dieses Rechteck trifft, soll sich der Knilch automatisch kurz drehen und die Nuss fangen.

Dies ist wichtig, um die Spielfreude nicht zu dämpfen. Ohne die `hitarea2` könnte man die knapp hinter dem Knilch zu Boden fallenden Nüsse nicht mehr erwischen, da der Knilch bei einer Drehaktion selbst im Mittelpunkt des Drehzentrums bleibt. Somit kann die fallende Nuss `hitarea1` nicht treffen.

Wir benötigen ein weiteres Hitfeld für die Kollision des Knilchs mit den im Spiel vorhandenen Hindernissen. Dieses Rechteck positionieren wir nun so, dass es den Knilch überdeckt. Wir geben ihm den Instanznamen `hitarea3`.

Jetzt setzen wir die Alphawerte der drei Hitfelder auf null. Somit sind die Rechtecke im Spiel nicht sichtbar und erfüllen nur eine funktionale Aufgabe.

Ich habe die Alphawerte vorläufig jedoch auf 20 % gesetzt. So kann ich bis zum Ende des Projekts die Hitfelder sehen und beobachten, wie sich das Spiel verhält.

Jetzt zu der Frage, wieso wir `hitarea3` im Netz-MovieClip positioniert haben und nicht an einer anderen Stelle. Würden wir `hitarea3` nicht im Netz-MovieClip positionieren, müssten wir es in allen sieben Animationsphasen in mehreren Schlüsselbildern positionieren. Bei der Sprunganimation müssten wir `hitarea3` ebenfalls in Sprungrichtung bewegen. Dieses ersparen wir uns, da das Netz sich in der Animationsphase befindet und bei einem Sprung das Netz und somit auch hitarea3 mit bewegt wird.

Als Nächstes erstellen wir einen neuen MovieClip: KnilchAnim

Dieser MovieClip besteht aus sieben Frames. In jeden Frame positionieren wir die obigen Animationsphasen.

Im ersten Frame wird die Knilch-Wait-Animation positioniert. Dieser Frame erhält den Bildnamen Wait und eine `this.stop()`-Aktion, der letzte Frame natürlich die Knilch-Fall-Animation (und der Frame wird mit Fall benannt).

Alle sieben Instanzen erhalten denselben Instanznamen Anim. Das ist wichtig, damit wir später diese Hitfelder mit `this.Anim.Netz.hitarea1 bis hitarea3` ansprechen können.

Deshalb erhält auch jedes Vorkommen des MovieClips Netz in den Animationsphasen den Instanznamen `Netz`.

Schließlich erstellen wir den **Haupt-MovieClip** für den Knilch:

Im ersten Frame positionieren wir eine Instanz von KnilchAnim und geben ihr den Instanznamen `KnilchAnim`. Diesen Frame benennen wir mit `Right`. Im zweiten Frame spiegeln wir den Knilch horizontal, so dass er nach links zeigt. Diesen zweiten Frame benennen wir mit `Left`.

Diesen Haupt-MovieClip positionieren wir auf der Bühne in der Hauptzeitleiste oberhalb der ersten Spielebene L1 und geben ihm den Instanznamen `Player`.

Jetzt können wir den Knilch mit der Anweisung `_root.Player.gotoAndStop ("Left")` nach links zeigen und ihn mit `_root.Player.KnilchAnim.gotoAndStop ("Run")` laufen lassen.

Erstellung des Lochhindernisses

Dieses Hindernis ist das einfachste der drei Hindernisse im Spiel. Wir platzieren eine Grafik, die ein Loch darstellt, in der ersten Ebene des Spiels (L1). Über das Loch positionieren wir ein Rechteck, das wieder als Hitfeld benutzt wird. Wir geben ihm den Instanznamen `hitareaLoch`.

Wenn der Knilch dieses Feld trifft, wird die Fallanimation eingeleitet; der Knilch strampelt in dieser Animationsphase mit den Füßen, fällt in das Loch hinein und verliert dabei die Nüsse. Abschließend kommt er aus dem Loch heraus, und das Spiel geht weiter.

Erstellung des Krebshindernisses

Der Krebs besteht aus zwei Animationsphasen. Jede Phase kommt in einen eigenen MovieClip. Wir erstellen somit zwei MovieClips: Krebs-Wait und Krebs-Snap.

Als Nächstes erstellen wir den Haupt-MovieClip Krebs. Dieser Haupt-MovieClip besteht aus zwei Frames. In jeden Frame positionieren wir die obigen Animationsphasen. In den ersten Frame wird die Krebs-Wait-Animation positioniert. Dieser Frame erhält den Bildnamen `Wait` und eine `this.stop()`-Aktion.

◀ **Abbildung 2**
Und so sieht unser Coco-Catch aus. Vorne im Bild der Knilch am Loch-Hindernis. Er hat schon drei Kokosnüsse gefangen

Im zweiten Frame wird die Krebs-Snap-Animation positioniert. Dieser Frame erhält den Bildnamen `Snap` und eine `this.stop()`-Aktion.

Beide Instanzen erhalten den Instanznamen `Anim`. Das ist wichtig, damit wir später mit `this.Anim.hitareaSnap` die Hitfelder ansprechen können.

Wir benötigen zwei Hitfelder, um das Verhalten des Krebses steuern zu können. Die beiden Hitfelder werden in beiden Animationsphasen oberhalb des Krebses positioniert. Das erste Hitfeld ist viel breiter als das zweite. Es dient dazu, die nähere Umgebung des Krebses zu markieren. Wenn der Knilch diesen Bereich betritt, soll der Krebs von der Wait-Phase in die Snap-Phase wechseln und in einem Loop nach dem Knilch schnappen. Die Schnappanimation wird so lange wiederholt, bis der Knilch das erste Hitfeld verlässt oder das zweite betritt.

Das zweite Hitfeld markiert die Reichweite des Krebses. Betritt der Knilch dieses Feld, wird er gnadenlos vom Krebs gekniffen, stolpert daraufhin und verliert erneut seine Nüsse.

Wir geben dem ersten Hitfeld den Instanznamen `hitareaAwake` und dem zweiten `hitareaSnap`, wobei darauf geachtet werden muss, dass `hitareaSnap` oberhalb von `hitareaAwake` platziert wird, da sonst das größere `hitareaAwake` das `hitareaSnap` überdeckt.

Wir positionieren den Haupt-MovieClip in die erste Spielebene (L1) und geben ihm den Instanznamen `Krebs`.

Abbildung 3 ▶
Das Spinne-Hindernis

Erstellung des Spinnenhindernisses

Die Spinne besteht aus zwei Animationsphasen. In der ersten Phase ist die Spinne nicht animiert und verhält sich ruhig. Wenn der Knilch in ihr Umfeld eintritt, beginnt die Spinne warnend zu pulsieren.

Wir erstellen zunächst einen MovieClip.

Im ersten Frame positionieren wir die Spinne in ihrer Wait-Phase. Ab dem zweiten Frame beginnt die Pulsieranimation. Diese Animation erstellen wir aus zwei Einzelbildern der Spinne. Ein Loop über diese zwei Frames lässt die Spinne pulsieren. Wir platzieren ein Hitfeld über die Spinne und geben ihm den Instanznamen `hitareaBit`.

Diesen MovieClip positionieren wir in einen neuen Haupt-MovieClip und geben ihm den Instanznamen `Spinne`. Die Spinne wird in diesem Haupt-MovieClip in einem Dauerloop auf und ab bewegt, so als seile sie sich ständig auf und ab.

Im Haupt-MovieClip positionieren wir unterhalb der Spinne ein weiteres Hitfeld. Diesem zweiten Hitfeld geben wir den Instanznamen `hitareaPulse`.

Betritt der Knilch dieses Hitfeld, fängt die Spinne an zu pulsieren. Die Spinne pulsiert so lange, bis der Knilch das Hitfeld wieder verlässt oder von `hitareaBit` getroffen wird. In diesem Fall wird er gnadenlos von der Spinne gebissen. Daraufhin stolpert der Knilch – und verliert alle Nüsse.

Genauso wie bei dem Krebs müssen wir darauf achten, dass das `hitareaPulse` das kleinere `hitareaBit` nicht überdeckt. `hitareaPulse` muss also im Haupt-MovieClip auf der untersten Ebene der Zeitleiste positioniert werden.

Wir positionieren zwei Instanzen des Haupt-MovieClips in der ersten Spielebene (L1) und geben ihnen die Instanznamen `Spinne1` und `Spinne2`.

Erstellung der Affen und der Nüsse

Im Spiel befinden sich zwölf Affen, die aus den Palmen und Felsen der ersten Spielebene auftauchen, eine Nuss werfen und sich wieder verkriechen.

Die Affen bestehen aus drei Animationsphasen:

1. Aus dem Baum auftauchen
2. Wurfanimation durchführen
3. Verschwinden

Die Wurfanimation kommt in einen MovieClip, in dem der Affe eine Nuss in der Hand hält, die er werfen soll. Dieser MovieClip hat eine `this.stop()`-Aktion im ersten Frame.

Im zweiten Frame beginnt die Wurfanimation. Dieser Frame wird Wurf benannt. Wir erstellen jetzt nacheinander zwölf neue MovieClips:

In jedem dieser MovieClips erscheint ein Affe an einer anderen Stelle und verschwindet dann wieder. Die Affen in diesen zwölf MovieClips sind alle Instanzen des Wurfanimation-MovieClips. Alle Instanzen erhalten den Instanznamen `Affe`.

Dann benötigen wir für die Wurfanimation zwölf weitere MovieClips:

In jedem dieser MovieClips animieren wir den Fall einer Nuss. Jede Nuss nimmt eine andere Flugrichtung und ist einem der zwölf Affen zugeordnet (die Zuordnung geschieht im Code mittels ID).

Wir positionieren diese 24 MovieClips in die erste Spielebene (L1) und geben ihnen nacheinander die Instanznamen `Affe1, Nuss1, Affe2, Nuss2, ... , Affe12 und Nuss12.`

Also gibt `Affe1` in seiner Wurfanimation `Nuss1` den Befehl zu starten usw. Eine komplette Wurfanimationsphase läuft wie folgt ab (Beispiel für `Affe1`):

`Affe1` erhält den Befehl zu starten. Er kommt aus seinem Versteck heraus und gibt der Instanz `Affe` den Befehl `this.Affe.gotoAndPlay ("Wurf");`.Während der Wurfanimation erhält `Nuss1` den Befehl zu starten: `_root.L1.Nuss1.gotoAndPlay("Start").` Am Ende der Wurfanimation gibt die Instanz `Affe` seinem Parent `Affe1` den Befehl, sich zu verkrie-

chen: `_parent.gotoAndPlay("Hide").` `Affe1` verkriecht sich daraufhin. Wenn `Nuss1` zur Boden fällt oder vom Knilch gefangen wird, wird `Affe1` für einen neuen Wurf freigegeben. Die Implementierung folgt später.

Erstellung der Fahrzeuge

Im Spiel haben wir vier verschiedene Fahrzeuge, die sich von einem Spielrand zum anderen bewegen und ebenfalls Nüsse werfen. Jedem dieser Fahrzeuge ordnen wir mehrere Nüsse zu. Genauso wie die Affen können die Fahrzeuge die ihnen zugeordneten Nüsse starten. Auf der Bühne ist zu jeder Zeit nur eines der vier Fahrzeuge aktiv.

Den vier Fahrzeugen werden die Nüsse `Nuss13` bis `Nuss22` zugeordnet. Die Nüsse und Fahrzeuge kommen ebenfalls in die erste Spielebene L1.

Implementierung der Objekte (der Code zur Steuerung des Spiels)

Jetzt kommen wir zum interessanten Teil dieses Kapitels. Wir haben die nötigen MovieClips geschaffen und müssen diesen noch Leben einhauchen. Diesen Part erledigen wir mit Hilfe von objektorientierter Programmierung.

Wir benötigen zehn verschiedene Objekte (siehe »Überblick über Objekte und Aktionen«). Wir definieren zunächst alle Konstruktorfunktionen der Objekte. Mit dieser Vorgehensweise haben wir alle Eigenschaften der Objekte im Überblick. Danach implementieren wir die Methoden der Objekte. Zuletzt erzeugen wir mit dem `new`-Operator und der Konstruktorfunktion die benötigten Instanzen.

Konstruktorfunktion des Objekts ObjScroll

Dieser Konstruktor benötigt zur Initialisierung einiger Eigenschaften des `ObjScroll`-Objekts einige Parameter.

```
function ObjScroll (ScrollSpeed, ScrollOffset1, ScrollOffset2, Max-
Left, MaxRight) {
```

Erläuterung der Eigenschaften:

Speed ist die Eigenschaft, welche die Scrollgeschwindigkeit der ersten Spielebene L1 angibt. Speed wird mit dem Wert des Parameters `ScrollSpeed` initialisiert.

```
this.Speed = ScrollSpeed;
```

`Offset1` und `Offset2` beeinflussen die Scrollgeschwindigkeit der zweiten und dritten Spielebene (L2 und L3). L2 wird um den Wert von `Offset1` langsamer gescrollt, L3 um den Wert von `Offset2` (siehe Methode `SetScrollVal`).

```
this.Offset1 = ScrollOffset1;
this.Offset2 = ScrollOffset2;
```

Ist `ScrollAvail = true`, wird bei Bedarf gescrollt. Wenn der Knilch in eine Falle gerät, wird `ScrollAvail` auf `false` gesetzt. Mit einer IF-Abfrage wird somit verhindert, dass weiter gescrollt wird.

```
this.ScrollAvail = true;
```

`MaxLeft` und `MaxRight` geben den äußersten Rand des Spielfeldes an. Mit diesen beiden Eigenschaften wird verhindert, dass unendlich gescrollt wird. Erreicht L1 diese Grenzen, wird nicht weiter gescrollt (siehe `HitMaxLeft/-Right`).

```
this.MaxLeft = MaxLeft;
this.MaxRight = MaxRight;
```

`HitLeft` wird `true,` sobald der Knilch `hitareaLeft` auf der Bühne trifft. Dann wird in die entgegengesetzte Richtung nach rechts gescrollt (`HitRight` synonym).

```
this.HitLeft = false;
this.HitRight = false;
```

`HitMaxLeft` wird `true,` sobald der äußerste linke Rand des Spielfeldes `MaxLeft` erreicht ist. Dann muss das Scrollen stoppen, und der Knilch bleibt stehen (`HitMaxRight` synonym).

```
    this.HitMaxLeft = false;
    this.HitMaxRight = false;
```

Die X-Koordinaten der drei Spielebenen werden initialisiert:

```
    this.L1X = _root.L1._x;
    this.L2X = _root.L2._x;
    this.L3X = _root.L3._x;
}
```

Konstruktorfunktion des Objekts ObjKnilch

```
function ObjKnilch () {
```

Erläuterung der Eigenschaften:

`Mover` erhält den Wert von `TestMover`, wenn die neu errechnete Position des Knilchs gültig ist. Sonst behält `Mover` seinen alten Wert, und der Knilch wird nicht bewegt. Dies tritt ein, wenn gescrollt wird. Während der Scrollaktion befindet sich der Knilch in der Rennanimationsphase am Bildschirmrand. Wenn gescrollt wird, bleibt die X-Position des Knilchs die gleiche.

Statt der Neupositionierung des Knilchs wird die Bühne gescrollt.

```
    this.Mover = _root.Player._x;
    this.TestMover = this.Mover;
```

`MoveVal` enthält den Beschleunigungs-/Bremswert des Knilchs. Dies wird in den Methoden des Knilchs erläutert. Aus `MoveVal` errechnen sich `TestMover` und somit auch `Mover`.

```
    this.MoveVal = 0;
```

Ist `KnilchAvail=true`, ist es dem Knilch erlaubt, sich zu bewegen oder in die Animationsphase zu wechseln. Ist es `false`, wird damit verhindert, dass der Knilch in die nächste Animationsphase wechselt. Ein Animationsphasenwechsel ist dann erlaubt, wenn der Knilch sich nicht in der Jump-, Deliver-, Turn-, Stumble- oder Fallphase befindet.

Tritt der Knilch in eine dieser Phasen, wird `KnilchAvail false` und ein Phasenwechsel verhindert, außer beim Übergang von Jump- in die

Stumble- oder Fallphase. In so einem Fall tritt die Stumble- oder Fallphase sofort ein, und es wird nicht erst abgewartet, bis der Knilch mit beiden Füßen wieder auf dem Boden steht. Sonst wird ein Phasenwechsel erst dann wieder erlaubt, wenn die aktuelle Phase beendet und `KnilchAvail` auf `true` gesetzt wurde.

```
this.KnilchAvail = true;
```

Ist `CatchAvail true`, ist es dem Knilch erlaubt, Nüsse zu fangen. Ist es `false`, wird damit verhindert, dass herunterfallende Nüsse im Netz des Knilchs landen, solange er sich z. B. in der Stolperanimationsphase befindet.

```
this.CatchAvail = true;
```

`MouseX` gibt die Position des MovieClips Mouse zurück. In dem MovieClip befindet sich eine unsichtbare Taste, die ständig erreichbar sein muss. Deshalb verfolgt sie die Maus (`_root.Mouse.startDrag (true);`). Mit dieser Taste wird die Sprung- oder Nussabgabe-Animationsphase eingeleitet. Die Taste besitzt den Befehl `on (press) { _root.myKnilch.JumpOrDeliver(); }`

```
this.MouseX = _root.Mouse._x;
```

Die x-Koordinate des Knilchs in der Hauptzeitleiste und in der ersten Spielebene L1:

```
this.X = _root.Player._x;
this.PL1X = this.X-_root.myScroll.L1X;
```

`DiffX` ist die Entfernung zwischen der Maus und dem Knilch. Je größer die Entfernung, desto stärker ist die Beschleunigung in Richtung der Maus.

```
this.DiffX = this.MouseX-this.X;
```

LR nimmt die Werte Left und Right an, je nachdem, ob der Knilch nach links oder nach rechts zeigt/läuft. Der Wert von LR wird aus dem Wert von `DiffX` abgeleitet. Ist `DiffX` negativ, so befindet sich die Maus links vom Knilch, und LR erhält den Wert Left. Für `DiffX` ist dies positiv synonym. LR wird mit einem beliebigen String initialisiert.

```
    this.LR = "%L";
```

Jedes Mal, bevor der Wert von LR geändert wird, wird der alte Wert in LRLast gespeichert. Ohne LRLast geht uns die Information, ob ein Richtungswechsel eingetreten ist, verloren. Wenn der Fall eintritt, dass LR ungleich LRLast ist, wissen wir, dass ein Richtungswechsel stattgefunden hat und der Knilch sich in die andere Richtung drehen muss.

```
    this.LRLast = "%R";
```

`NetzNutCount` ist die Anzahl der Nüsse, die der Knilch in seinem Netz mit sich trägt. Der Knilch kann höchstens drei Nüsse gleichzeitig in seinem Netz mit sich tragen. Kommt eine vierte Nuss hinzu, fallen ihm alle Nüsse zu Boden, und `NetzNutCount` wird auf null gesetzt.

```
    this.NetzNutCount = 0;
}
```

Konstruktorfunktion des Objekts ObjKrebs

```
function ObjKrebs () {
```

Erläuterung der Eigenschaften:

Ist `SnapAvail true`, kann der Krebs den Knilch zum Stolpern bringen. `SnapAvail` wird `false`, sobald der Knilch durch den Krebs zum Stolpern gebracht wurde. `SnapAvail` wird erst dann wieder `true`, wenn die Stolperphase des Knilchs beendet ist und danach ein paar Sekunden vergangen sind.

```
    this.SnapAvail = true;
```

Ist `AwakeAvail true` und `HitAwakeArea` ebenfalls, wechselt der Krebs von der Wait-Phase zur Snap-Phase. Findet dieser Wechsel statt, wird `AwakeAvail false`. Es wird erst dann wieder `true`, wenn der Knilch `hitareaAwake` verlässt.

Die Notwendigkeit dieser Eigenschaft ist durchaus begründet. Ihr Fehlen würde den Krebs wiederholt von der Wait-Phase zur Snap-Phase wechseln lassen, und die Snap-Phase könnte nicht abgespielt werden.

```
    this.AwakeAvail = true;
```

HitSnapArea wird true, sobald der Knilch mit seinem hitarea3 das Hitfeld /hitareaSnap des Krebses berührt. (Siehe Erstellung des Krebshindernisses und Erstellung des Knilchs.)

```
    this.HitSnapArea = false;
```

HitAwakeArea wird true, sobald der Knilch mit seinem hitarea3 das Hitfeld hitareaAwake des Krebses berührt. (Siehe Erstellung des Krebshindernisses und Erstellung des Knilchs.)

```
    this.HitAwakeArea = false;
}
```

Konstruktorfunktion des Objekts ObjLoch

```
function ObjLoch () {
```

Erläuterung der Eigenschaften:

Ist LochAvail true, kann der Knilch in das Loch fallen. LochAvail wird false, sobald der Knilch in das Loch fällt. LochAvail wird erst dann wieder true, wenn die Fallanimationsphase des Knilchs beendet ist und danach ein paar Sekunden vergangen sind.

```
    this.LochAvail = true;
```

HitLochArea wird true, sobald der Knilch mit seinem hitarea3 das Hitfeld hitareaLoch des Lochs berührt. (Siehe Erstellung des Lochhindernisses und Erstellung des Knilchs.)

```
    this.HitLochArea = false;
}
```

Konstruktorfunktion des Objekts ObjSpinne

Dieser Konstruktor benötigt zur Initialisierung einen wichtigen Parameter namens ID. Von diesem Objekt werden später zwei Instanzen erzeugt, da wir zwei Spinnen auf der Bühne haben (Spinne1 und Spinne2). Diese beiden Spinnen teilen sich diesen Code. Damit die Methoden des Objekts

ObjSpinne wissen, auf welche der zwei Spinnen auf der Bühne sie sich beziehen müssen, ist diese ID-Eigenschaft nötig.

Wird diese Konstruktorfunktion mit 1 als Wert für ID aufgerufen, bezieht sich der Code der neuen Instanz auf _root.L1.Spinne1; mit 2 als Wert für die ID synonym.

```
function ObjSpinne (ID) {
```

Erläuterung der Eigenschaften:

Identifikator der zu erzeugenden Spinnenobjekte und der Spinnen auf der Bühne.

```
   this.ID = ID;
```

Synonym zu SnapAvail des Krebses.

```
   this.BitAvail = true;
```

Synonym zu AwakeAvail des Krebses.

```
   this.PulseAvail = true;
```

Synonym zu HitSnapArea des Krebses.

```
   this.HitBitArea = false;
```

Synonym zu HitAwakeArea des Krebses.

```
   this.HitPulseArea = false;
```

Das ID des Objektes wird an die Spinne (mit dem Wert der ID als Namensende) weitergegeben. Ist ID = 1, wird die Variable _root.L1.Spinne1.ID = 1 gesetzt, für ID = 2 wird die Variable _root.L1.Spinne2.ID = 2 gesetzt.

Damit können die Spinnen ihrerseits auf die richtige Instanz dieses Objekts zugreifen. Spinne1 kann mit _root["mySpinne"+this.ID].PulseAvail den Wert der Eigenschaft PulseAvail von mySpinne1 abfragen.

```
   _root.L1["Spinne"+this.ID].ID = this.ID;
}
```

Konstruktorfunktion des Objekts ObjTime

Die Spielzeit wird in Minuten und Sekunden dem Konstruktor als Parameter übergeben.

```
function ObjTime (Min, Sec) {
```

Erläuterung der Eigenschaften:
Spielminuten und Sekunden werden initialisiert.

```
   this.Min = Min;
   this.Sec = Sec;
```

Spielzeit in Sekunden; wird `TotalSec` == 0, ist die Spielzeit zu Ende.

```
   this.TotalSec = 60*this.Min+this.Sec+1;
```

`Timer` ist ein String, der aus den Werten Min und Sec generiert wird. Er wird auf dem Bildschirm als verbleibende Spielzeit angezeigt.

```
   this.Timer = "";
}
```

Konstruktorfunktion des Objekts ObjScore

```
function ObjScore () {
```

Erläuterung der Eigenschaften:
`TotalScore` ist die erreichte Punktzahl.

```
   this.TotalScore = 0;
```

`NetzNutPoints` ist die Punktezahl der Nüsse, die der Knilch in seinem Netz trägt. Gibt er die Nüsse an der Bar ab, wird dieser Wert zu `TotalScore` addiert und anschließend auf null gesetzt. Gerät der Knilch in eine Falle und verliert die Nüsse im Netz, wird dieser Wert auf null gesetzt. Dies ist ebenfalls so, wenn der Knilch eine vierte Nuss in seinem Netz fängt.

```
   this.NetzNutPoints = 0;
```

Die Werte von `TotalScore` und `NetzNutPoints` werden auf dem Bildschirm angezeigt. Diese Anzeigen werden zur Beginn mit null initialisiert.

```
   _root.Display.NetzNutPoints = 0;
   _root.Display.TotalScore = 0;
}
```

Konstruktorfunktion des Objekts ObjNuss

Dieser Konstruktor benötigt zur Initialisierung einen wichtigen Parameter namens ID. Von diesem Objekt werden später 22 Instanzen erzeugt, da wir 22 Nüsse auf der Bühne haben (Nuss1 bis Nuss22: siehe Erstellung der Affen und Nüsse). Diese 22 Nüsse teilen sich den Code. Damit die Methoden des `ObjNuss`-Objekts wissen, auf welche der 22 Nüsse auf der Bühne sie sich beziehen müssen, ist diese ID-Eigenschaft nötig. Wird diese Konstruktorfunktion mit 1 als Wert für ID aufgerufen, bezieht sich der Code der neuen Instanz auf `_root.L1.Nuss1;` Mit 2 bis 22 als Wert für ID ist es synonym.

```
function ObjNuss (ID, NutPoints) {
```

Erläuterung der Eigenschaften:
Identifikator der zu erzeugenden Nussobjekte und der Nüsse auf der Bühne.

```
   this.ID = ID;
```

Jede Nuss auf der Bühne erzielt eine andere Punktezahl. Diese Punktezahl wird in dieser Eigenschaft des Objekts `ObjNuss` gespeichert.

```
   this.NutPoints = NutPoints;
```

Das ID des Objektes wird an die Nuss (mit Wert von ID als Namensende) weitergegeben.

Ist ID = 1, wird die Variable `_root.L1.Nuss1.ID = 1` gesetzt, für ID = 22 wird die Variable `_root.L1.Nuss22.ID = 22` gesetzt.

Damit können die Nüsse ihrerseits auf die richtige Instanz dieses Objekts `ObjNuss` zugreifen. Nuss1 kann mit `_root["myNuss"+this.ID].PulseAvail` den Wert von `NutPoints` von `myNuss1` abfragen.

```
    _root.L1["Nuss"+this.ID].ID = this.ID;
}
```

Konstruktorfunktion des Objekts ObjAffeGen

Dieser Konstruktor benötigt zur Initialisierung einige Parameter.

```
function ObjAffeGen (RandomAffeVal, AffenActiveMax, AffenCount) {
```

Erläuterung der Eigenschaften (siehe Methode von `ObjAffeGen`):

```
    this.RandomAffeVal = RandomAffeVal;
```

Ein Wert für das zufällige Starten einer der zwölf Affen auf der Bühne.

```
    this.RAffe = 0;
```

Dies bezeichnet die Anzahl der Affen, die sich in der Wurfphase befinden; eine Wurfphase gilt erst dann als beendet, wenn die zugehörige Nuss zu Boden fällt oder gefangen wurde. (Siehe Erstellung der Affen und Nüsse.)

Dieser Wert wird erhöht, sobald ein neuer Affe startet, und vermindert, sobald die Wurfphase eines Affen beendet ist.

```
    this.AffenActiveCount = 0;
```

Anzahl der Affen, die gleichzeitig aktiv sein dürfen. Ist `this.AffenActiveCount` größer als `this.AffenActiveMax`, wird das Starten eines neuen Affen so lange verhindert, bis die Bedingung `true` wird.

```
    this.AffenActiveMax = AffenActiveMax;
```

`AffenCount` ist einfach die Anzahl aller Affen auf der Bühne, also zwölf.

```
    this.AffenCount = AffenCount;
```

`AffeAvail` ist ein Array, das von 1 bis 12 mit `false` initialisiert wird. `AffeAvail[]` dient als Sperre. Beispiel: Wurde `Affe1` gestarten, wird `AffeAvail[1]` sofort auf `true` gesetzt. Solange dieser Wert `true` ist, wird `Affe1` nicht erneut von dem Code gestartet. `AffeAvail[1]` wird erst dann wieder `false`, wenn `Nuss1` zu Boden fällt oder gefangen wurde. Für `AffeAvail[2]` bis `AffeAvail[12]` synonym.

```
   this.AffeAvail = new Array();
   for (i=1; i<this.AffenCount+1; i++) {
      this.AffeAvail[i] = false;
```

Genauso wie die Nüsse erhält Affe1 ID 1 und Affe12 ID 12. Mit Hilfe dieser ID kann jeder Affe die ihm zugeordnete Nuss identifizieren.

```
      _root.L1["Affe"+i].ID = i;
   }
```

Ein Array wird beginnend mit null indiziert. Um Unübersichtlichkeiten zu vermeiden, haben wir begonnen, `AffeAvail[]` beginnend mit 1 zu initialisieren, da Affe1 unser erster Affe ist. Sicherheitshalber initialisieren wir das erste Element des Arrays auch mit `false`. Das Feld `AffeAvail[0]` wir jedoch nie benutzt.

```
   this.AffeAvail[0] = false;
}
```

Konstruktorfunktion des Objekts ObjMobileGen

Dieser Konstruktor benötigt zur Initialisierung einen Parameter.

```
function ObjMobileGen (MobileForms) {
```

Erläuterung der Eigenschaften:
Wir haben vier verschiedene Fahrzeugvarianten.

```
   this.MobileForms = MobileForms;
```

Zu jeder Zeit darf nur ein Fahrzeug aktiv sein. Sobald ein Fahrzeug gestartet wird, wird dieser Wert auf `true` gesetzt. Ist dieser Wert `true`, wird so-

mit verhindert, dass ein neues Fahrzeug vom Code gestartet wird, solange ein anderes bereits aktiv ist.

Dieser Wert wird erst dann wieder `false`, wenn das aktive Fahrzeug am Ende ankommt und von der Bühne verschwindet.

```
    this.MobileInUse = false;
```

Die vier verschiedenen Fahrzeuge tauchen in einem Loop auf. `MobileNum` gibt an, welches der vier Fahrzeuge an der Reihe ist.

```
    this.MobileNum = 0;
```

Mit `RandomMobile` wird per Zufall entschieden, ob das Fahrzeug von links oder von rechts kommen soll.

```
    this.RandomMobile = 0;
```

LR wird von `RandomMobile` abgeleitet.

```
    this.LR = "%";
}
```

Alle Konstruktorfunktionen der benötigten Objekte sind jetzt definiert. Nun sind uns alle Eigenschaften der Objekte bekannt. Wir müssen noch die Methoden der Objekte definieren. Hierfür definieren wir, wie im Kapitel über objektorientierte Programmierung gezeigt, die benötigten Funktionen.

Diese Funktionen werden wir dann der Prototype-Eigenschaft des zugehörigen Objekts zuweisen. Somit werden die Funktionen zu Methoden der obigen Objekte.

Methoden

Methoden des ObjScroll-Objekts

Mit dieser Methode wird die zuvor deaktivierte Scrollfunktion wieder aktiviert. Diese Methode wird z. B. am Ende der Stolperanimation des Knilchs mit `_root.myScroll.StartScroll();` aufgerufen:

```
function _Scroll_StartScroll_ () {
```

Aufhebung der Sperren:

```
    _root.myKnilch.CatchAvail = true;
    _root.myScroll.ScrollAvail = true;
    _root.myKnilch.KnilchAvail = true;
}
```

Mit dieser Methode wird die Scrollfunktion deaktiviert. Diese Methode wird z. B. zu Beginn der Stolperanimation des Knilchs mit `_root.myScroll.StopScroll();` aufgerufen.

```
function _Scroll_StopScroll_ () {
```

Sperre der zu sperrenden Eigenschaften (siehe Erläuterungen der Eigenschaften der zugehörigen Konstruktorfunktionen).

```
   this.ScrollAvail = false;
   _root.myKnilch.KnilchAvail = false;
   _root.myKnilch.CatchAvail = false;
   _root.SKnilchRun.gotoAndStop(1);
}
```

Mit dieser Methode werden die Eigenschaften L1X, L2X und L3X neu gesetzt.

```
function _Scroll_GetLayerX_ () {
   this.L1X = _root.L1._x;
   this.L2X = _root.L2._x;
   this.L3X = _root.L3._x;
}
```

Mit dieser Methode werden die X-Positionen der drei Spielebenen aktualisiert.

```
function _Scroll_SetLayerX_ () {
   _root.L1._x = this.L1X;
   _root.L2._x = this.L2X;
   _root.L3._x = this.L3X;
}
```

Mit dieser Methode werden die Hiteigenschaften des Objekts neu gesetzt.

```
function _Scroll_TestScroll_ () {
```

Ist der Knilch am linken oder rechten Rand angekommen?

```
   this.HitLeft = _root.Player.hitTest(_root.hitareaLeft);
   this.HitRight = _root.Player.hitTest(_root.hitareaRight);
```

Ist der äußerste linke oder rechte Rand des Spielfeldes erreicht?

```
   this.HitMaxLeft = this.L1X>this.MaxLeft;
   this.HitMaxRight = this.L1X<this.MaxRight;
}
```

Mit dieser Methode werden die neuen Positionen der drei Spielebenen berechnet. Spielebene2 wird relativ zur Spielebene1 langsamer gescrollt. Spielebene3 wird relativ zur Spielebene2 langsamer gescrollt.

```
function _Scroll_SetScrollVal_ (vSpeed, vOffset1, vOffset2) {
   this.L1X = this.L1X+vSpeed;
   this.L2X = this.L2X+vSpeed+vOffset1;
   this.L3X = this.L3X+vSpeed+vOffset2;
}
```

Mit dieser Methode wird das Scrollen durchgeführt. Es ist die Hauptmethode des Objekts `ObjScroll`.

```
function _Scroll_SetScroll_ () {
   this.GetLayerX();
   this.TestScroll();
```

Wenn der Knilch am rechten Rand angekommen ist, der äußerste rechte Rand nicht erreicht und das Scrollen nicht gesperrt ist, dann scrolle nach links.

```
   if (this.HitRight && !this.HitMaxRight && this.ScrollAvail) {
      this.SetScrollVal(-this.Speed, this.Offset1, this.Offset2);
      this.SetLayerX();
```

Wenn der Knilch am linken Rand angekommen ist, der äußerste linke Rand nicht erreicht und das Scrollen nicht gesperrt ist, dann scrolle nach rechts.

```
   } else if (this.HitLeft && !this.HitMaxLeft && this.ScrollAvail) {
      this.SetScrollVal(this.Speed, -this.Offset1, -this.Offset2);
      this.SetLayerX();
   }
}
```

Die obigen Funktionen werden der Prototype-Eigenschaft des Objekts `ObjScroll` zugewiesen und werden somit zu dessen Methoden.

```
ObjScroll.prototype.StartScroll = _Scroll_StartScroll_;
ObjScroll.prototype.StopScroll = _Scroll_StopScroll_;
ObjScroll.prototype.GetLayerX = _Scroll_GetLayerX_;
ObjScroll.prototype.SetLayerX = _Scroll_SetLayerX_;
ObjScroll.prototype.TestScroll = _Scroll_TestScroll_;
ObjScroll.prototype.SetScrollVal = _Scroll_SetScrollVal_;
ObjScroll.prototype.SetScroll = _Scroll_SetScroll_;
```

Methoden des ObjKnilch-Objekts

Mit dieser Methode werden die Bewegungseigenschaften neu ermittelt und gesetzt.

```
function _Knilch_SetMoveParams_ () {
```

Siehe Erläuterungen der Eigenschaften der zugehörigen Konstruktorfunktion.

Neue Position der Maus und des Knilchs ermitteln.

```
   this.MouseX = _root.Mouse._x;
   this.X = _root.Player._x;
   this.PL1X = this.X-_root.myScroll.L1X;
```

Neue Entfernung zwischen Maus und Knilch ermitteln.

```
   this.DiffX = this.MouseX-this.X;
```

LR wird in LRLast gespeichert und der neue Wert aus `DiffX` abgeleitet.

```
   this.LRLast = this.LR;
   if (this.DiffX<0) {
      this.LR = "Left";
   } else if (this.DiffX>0) {
      this.LR = "Right";
   }
```

Formel zur beschleunigten Bewegung des Knilchs; der Beschleunigungs-Abbremswert wird aus `DiffX` abgeleitet. `MoveVal` wird in Abhängigkeit von `DiffX` positiv oder negativ. Ist es negativ, so wird der Knilch nach links, sonst nach rechts bewegt.

Die Konstanten 15 und 85 wurden nach langem Experimentieren so gewählt; ich musste so lange damit spielen, bis das gewünschte Verhalten zu beobachten war, wobei das gewünschte Verhalten ebenfalls unbekannt war.

```
   this.MoveVal = int((this.MoveVal+15*this.DiffX)/85);
}
```

Mit dieser Methode wird die Bewegung des Knilchs gesteuert.

```
function _Knilch_MoveKnilch_ () {
```

Nur wenn das Scrollen erlaubt ist, wird die neue Position ermittelt.

```
   if (_root.myScroll.ScrollAvail) {
```

`Mover` ist die aktuelle Position und `MoveVal` der errechnete beschleunigte Offset-Wert.

```
      this.TestMover = this.Mover+this.MoveVal;
   }
```

Wenn der Knilch weder die rechte noch die linke Seite erreicht hat, wird `Mover` auf `TestMover` gesetzt, und der Knilch wird bewegt. Der Ausdruck in dieser IF-Anweisung ist nach DeMorgan äquivalent zur `! _root.myScroll.HitLeft && ! _root.myScroll.HitRight`.

Tipp: Grundkenntnisse in der Boole´schen Algebra können helfen, Ausdrücke schnell zu konstruieren und sehr komplexe Ausdrücke zu optimieren.

```
if (!(_root.myScroll.HitLeft || _root.myScroll.HitRight)) {
   this.Mover = this.TestMover;
   _root.Player._x = this.Mover;
}
```

Wenn ein Animationsphasenwechsel erlaubt ist, dann ...

```
if (this.KnilchAvail) {
```

Ist ein Richtungwechsel aufgetreten? Wenn ja, starte die Drehanimationsphase.

```
   if (this.LRLast != this.LR) {
      this.KnilchAvail = false;
      _root.Player.KnilchAnim.gotoAndStop("Turn");
```

Kein Richtungswechsel! Wenn folgende drei Bedingungen gelten, darf der Knilch in die Rennphase wechseln oder sich darin aufhalten:

1. Ist der Beschleunigungswert kleiner als –1 oder größer 1?
2. Hat der Knilch den linken Spielrand nicht erreicht? Oder, wenn er ihn schon erreicht hat: Ist der äußerste linke Spielfeldrand noch nicht erreicht?
3. Hat der Knilch den rechten Spielrand nicht erreicht? Oder, wenn er ihn schon erreicht hat: Ist der äußerste rechte Spielfeldrand noch nicht erreicht?

Wenn alle drei Bedingungen zutreffen, dann starte Rennphase oder verbleibe in der Rennphase.

Die drei Bedingungen wurden wie folgt formuliert und sind äquivalent zu der Bedingung im Code: `(this.MoveVal>1 || this.MoveVal<-1) && (! _root.myScroll.HitLeft || ! _root.myScroll.HitMaxLeft) && (! _root.myScroll.HitRight || ! _root.myScroll.HitMaxRight)`
Jetzt geht es weiter mit dem Listing:

```
   } else if ((this.MoveVal>1 || this.MoveVal<-1) &&
         !(_root.myScroll.HitMaxLeft &&
           _root.myScroll.HitLeft) &&
```

```
            !(_root.myScroll.HitMaxRight &&
              _root.myScroll.HitRight)) {
         _root.Player.gotoAndStop(this.LR);
         _root.Player.KnilchAnim.gotoAndStop("Run");
         _root.SKnilchRun.gotoAndPlay("Sound");
```

Zur Übung:

Sonst ist eine dieser Bedingungen `true`:

1. `MoveVal` ist nahe null, oder
2. der Knilch hat den linken Spielfeldrand erreicht, oder
3. der Knilch hat den rechten Spielfeldrand erreicht.

Dieser Ausdruck ist die Negation der zweiten IF-Anweisung. Somit ist

```
!((this.MoveVal<=1 && this.MoveVal>=-1) ||
(_root.myScroll.HitLeft && _root.myScroll.HitMaxLeft) ||
(_root.myScroll.HitRight && _root.myScroll.HitMaxRight))
```

ebenfalls äquivalent zur obigen IF-Anweisung.

In diesem letzten Fall wird die Warteanimation gestartet.

```
      } else {
         _root.Player.KnilchAnim.gotoAndStop("Wait");
         _root.SKnilchRun.gotoAndStop(1);
      }
   }
}
```

Mit dieser Methode wird die Sprung- oder Nussabgabeanimation des Knilchs gestartet. Diese Methode wird per Mausklick aufgerufen.

```
function _Knilch_JumpOrDeliver_ () {
```

Falls mehr als null Nüsse im Netz des Knilchs sind und der Knilch sich an der Bar befindet, soll er die Nüsse an der Bar abgeben, anstatt zu springen.

```
   if (this.NetzNutCount>0 &&_root.L1.hitareaKiosk.hitTest
         (_root.Player.KnilchAnim.Anim.Netz.hitarea3)) {
         _root.myScroll.StopScroll();
         _root.Player.KnilchAnim.gotoAndStop("Deliver");
```

Eine Methode des `ObjScore`-Objekts wird aufgerufen, welche die Punkte im Netz zu `TotalScore` addiert; siehe unten.

```
        _root.myScore.SetTotalScore();
```

Ist der Knilch aber von der Bar entfernt oder hat keine Nüsse im Netz, soll er springen.

```
   } else {
        this.KnilchAvail = false;
        _root.SKnilchRun.gotoAndStop(1);
        _root.SKnilchJump.gotoAndPlay("Sound");
        _root.Player.KnilchAnim.gotoAndStop("Jump");
   }
}
```

Die obigen Funktionen werden der Prototype-Eigenschaft des `ObjKnilch`-Objekts zugewiesen und werden somit zu dessen Methoden.

```
ObjKnilch.prototype.SetMoveParams = _Knilch_SetMoveParams_;
ObjKnilch.prototype.MoveKnilch = _Knilch_MoveKnilch_;
ObjKnilch.prototype.JumpOrDeliver = _Knilch_JumpOrDeliver_;
```

Methoden des ObjKrebs-Objekts

Mit dieser Methode wird geprüft, ob der Krebs den Knilch erwischt hat: Sie steuert also das Verhalten des Krebses.

```
function _Krebs_Check_ () {
```

Testet, ob der Knilch mit seinem `hitarea3` das `hitareaAwake` des Krebses berührt hat.

```
   this.HitAwakeArea = _root.L1.Krebs.Anim.hitareaAwake.hitTest
(_root.Player.KnilchAnim.Anim.Netz.hitarea3);
```

Testet, ob der Knilch mit seinem `hitarea3` das `hitareaSnap` des Krebses berührt hat.

```
   this.HitSnapArea = _root.L1.Krebs.Anim.hitareaSnap.hitTest
(_root.Player.KnilchAnim.Anim.Netz.hitarea3);
```

Nur wenn der Knilch nicht schon bereits in Krebsfalle gegangen ist:

```
   if (this.SnapAvail) {
```

Wenn der Knilch noch nicht das `hitareaAwake` des Krebses berührt hat und die Snap-Animation nicht schon bereits eingeleitet wurde, dann starte Snap-Phase.

```
      if (this.AwakeAvail && this.HitAwakeArea) {
```

Wichtig: sofortige Sperre, damit nicht ständig erneut die Snap-Phase gestartet wird.

```
         this.AwakeAvail = false;
         _root.L1.Krebs.gotoAndStop("Snap");
      }
```

Wenn der Knilch vom Krebs erwischt wurde, dann starte die Stolperphase des Knilchs.

```
      if (this.HitSnapArea) {
```

Wichtig: sofortige Sperre, damit nicht ständig erneut die Stolperphase gestartet wird.

```
         this.SnapAvail = false;
         this.AwakeAvail = false;
         _root.myScroll.StopScroll();
         _root.SKnilchStumble.gotoAndPlay("Sound");
         _root.Player.KnilchAnim.gotoAndStop("Stumble");
      }
   }
}
```

Die obige Funktion wird der Prototype-Eigenschaft des Objekts `ObjKrebs` zugewiesen und somit zu dessen Methode.

```
ObjKrebs.prototype.Check = _Krebs_Check_;
```

Methoden des ObjLoch-Objekts

Mit dieser Methode wird geprüft, ob der Knilch in das Loch gefallen ist.

```
function _Loch_Check_ () {
```

Testet, ob der Knilch mit seinem `hitarea3` das `hitareaLoch` des Lochs berührt hat.

```
   this.HitLochArea = _root.L1.Loch.hitareaLoch.hitTest
(_root.Player.KnilchAnim.Anim.Netz.hitarea3);
```

Wenn der Knilch nicht schon bereits das `hitareaLoch` des Lochs berührt hat und die Fallanimation nicht schon bereits eingeleitet wurde, dann starte Fallphase.

```
    if (this.LochAvail && this.HitLochArea) {
```

Wichtig: sofortige Sperre, damit nicht ständig erneut die Fallphase gestartet wird.

```
      this.LochAvail = false;
      _root.myScroll.StopScroll();
      _root.Player.KnilchAnim.gotoAndStop("Fall");
      _root.L1.Loch.gotoAndStop(_root.myKnilch.LR);
   }
}
```

Die obige Funktion wird der Prototype-Eigenschaft des Objekts `ObjLoch` zugewiesen und wird somit zu dessen Methode.

```
ObjLoch.prototype.Check = _Loch_Check_;
```

Methoden des ObjSpinne-Objekts

Mit dieser Methode wird geprüft, ob die Spinne[ID] den Knilch erwischt hat. Sie steuert das Verhalten der Spinne.

(Im Kommentar meinen wir mit Spinne[ID], Spinne1 für ID = 1 und Spinne2 für ID = 2; siehe Konstruktorfunktion des `ObjSpinne`-Objekts.)

```
function _Spinne_Check_ () {
```

Testet, ob der Knilch mit seinem `hitarea3` das `hitareaPulse` der Spinne[ID] berührt hat.

```
   this.HitPulseArea = _root.L1["Spinne"+this.ID]
.hitareaPulse.hitTest
(_root.Player.KnilchAnim.Anim.Netz.hitarea3);
```

Testet, ob der Knilch mit seinem `hitarea3` das `hitareaBit` der Spinne[ID] berührt hat.

```
   this.HitBitArea = _root.L1["Spinne"+this.ID].
Spinne.hitareaBit.hitTest
(_root.Player.KnilchAnim.Anim.Netz.hitarea3);
```

Nur, wenn der Knilch nicht schon bereits in die Spinne[ID]-Falle gegangen ist.

```
   if (this.BitAvail) {
```

Wenn der Knilch noch nicht das `hitareaPulse` der Spinne[ID] berührt hat und die Pulse-Animation nicht schon bereits eingeleitet wurde, dann starte Pulse-Phase.

```
      if (this.PulseAvail && this.HitPulseArea) {
```

Wichtig: sofortige Sperre, damit nicht ständig erneut die Pulse-Phase gestartet wird.

```
         this.PulseAvail = false;
         _root.L1["Spinne"+this.ID].Spinne.gotoAndPlay("Pulse");
      }
```

Wenn der Knilch von Spinne[ID] erwischt wurde, dann starte die Stolperphase des Knilchs.

```
      if (this.HitBitArea) {
```

Wichtig: sofortige Sperre, damit nicht ständig erneut die Fallphase gestartet wird.

```
         this.BitAvail = false;
         this.PulseAvail = false;
         _root.myScroll.StopScroll();
         _root.SSpinne.gotoAndPlay("Sound");
         _root.Player.KnilchAnim.gotoAndStop("Stumble");
      }
   }
}
```

Die obige Funktion wird der Prototype-Eigenschaft des Objekts `ObjSpinne` zugewiesen und wird somit zu dessen Methode.

```
ObjSpinne.prototype.Check = _Spinne_Check_;
```

Methoden des ObjTime-Objekts

Mit dieser Methode wird der Count-down implementiert und die Spielzeitanzeige aktualisiert.

```
function _Time_CountDown_ () {
```

`TotalSec` um eine Sekunde vermindern.

```
   this.TotalSec--;
```

Ermitteln der restlichen Spielminuten (ganzzahlige Division).

```
   this.Min = int(this.TotalSec/60);
```

Ermitteln der restlichen Spielsekunden (Rest der ganzzahligen Division).

```
   this.Sec = this.TotalSec%60;
```

Baut ein String für die Anzeige auf (Format: XX:XX).

```
   if (this.Sec>9) {
      this.Timer = this.Min+":"+this.Sec;
   } else {
      this.Timer = this.Min+":0"+this.Sec;
   }
```

Aktualisiert die Anzeige.

```
   _root.Display.Timer = this.Timer;
}
```

Mit dieser Methode wird überprüft, ob die Spielzeit abgelaufen ist.

```
function _Time_EndGame_ () {
```

Wenn Spielzeit abgelaufen ist, beende das Spiel.

```
   if (this.TotalSec == 0) {
      _root.StopSounds.gotoAndPlay(2);
      _root.gotoAndStop("Ende");
   }
}
```

Die obigen Funktionen werden der Prototype-Eigenschaft des `ObjTime`-Objekts zugewiesen und werden somit zu dessen Methoden.

```
ObjTime.prototype.CountDown = _Time_CountDown_;
ObjTime.prototype.EndGame = _Time_EndGame_;
```

Methoden des ObjScore-Objekts

Mit dieser Methode werden die Punkte im Netz des Knilchs zusammengezählt.

```
function _Score_AddNetzNutPoints_ (Points) {
```

Wird diese Methode mit dem Parameterwert 0 aufgerufen, wird die Punktezahl im Netz nicht akkumuliert, sondern auf null gesetzt. Diese Methode wird dann mit Parameterwert 0 aufgerufen, wenn der Knilch seine Nüsse, die er im Netz trägt, verliert.

```
   if (Points == 0) {
      this.NetzNutPoints = 0;
   } else {
      this.NetzNutPoints += Points;
   }
```

Aktualisiert die Anzeige der Netzpunkte.

```
   _root.Display.NetzNutPoints = this.NetzNutPoints;
}
```

Mit dieser Methode werden die Punkte im Netz zum `TotalScore` addiert. Diese Methode wird aufgerufen, wenn der Knilch Nüsse an der Bar abgibt.

```
function _Score_SetTotalScore_ () {
   this.TotalScore += this.NetzNutPoints;
   this.AddNetzNutPoints(0);
   _root.Display.TotalScore = this.TotalScore;
}
```

Die obigen Funktionen werden der Prototype-Eigenschaft des `ObjTime`-Objekts zugewiesen und werden somit zu dessen Methoden.

```
ObjScore.prototype.AddNetzNutPoints = _Score_AddNetzNutPoints_;
ObjScore.prototype.SetTotalScore = _Score_SetTotalScore_;
```

Methoden des ObjNuss-Objekts

Mit dieser Methode werden die Punkte im Netz des Knilchs zusammengezählt.

```
function _Nuss_AddPoints_ () {
```

Wenn der Knilch bereits drei Nüsse in seinem Netz trägt und jetzt eine vierte Nuss hinzukommen soll, wird es hier verhindert, da der Knilch nur drei Nüsse gleichzeitig tragen darf.

```
if (_root.myKnilch.NetzNutCount == 3) {
```

`OverFlow` ist eine kleine Animation. Man sieht, wie die Nüsse aus dem Netz heraus fallen. Diese Animation wird auf die X-Position des Knilchs in der ersten Spielebene L1 gesetzt und gestartet. Die Anzahl der Nüsse wird daraufhin auf null gesetzt; außerdem wird auch die Punkteanzeige für das Netz auf null gesetzt.

```
    _root.L1.OverFlow._x = _root.myKnilch.PL1X;
    _root.L1.OverFlow.gotoAndPlay("Loose");
    _root.myKnilch.NetzNutCount = 0;
    _root.myScore.AddNetzNutPoints(0);
```

Sonst ist das Netz noch nicht voll, und der Knilch kann die Nuss aufnehmen.

```
} else {
    _root.SNutCatch.gotoAndPlay("Sound");
```

Die Anzahl der Nüsse im Netz wird um eins erhöht, und die Punkteanzeige für das Netz wird um die der Nuss zugewiesene Punktezahl erhöht.

```
    _root.myKnilch.NetzNutCount++;
    _root.myScore.AddNetzNutPoints(this.NutPoints);
}
```

Wenn die gefangene Nuss eine `ID <= 12=AffenCount` hat, dann wurde diese Nuss von einem der Affen geworfen. In diesem Fall wird die Anzahl der aktiven Affen um eins vermindert (siehe Methode `DecAffe(ID)` von `ObjAffeGen`).

```
if (this.ID<_root.myAffeGen.AffenCount+1) {
    _root.myAffeGen.DecAffe(this.ID);
}
```

Die Nuss mit der richtigen ID in der ersten Spielebene L1 wird gestoppt, da sie ja gefangen wurde.

```
   _root.L1["Nuss"+this.ID].gotoAndStop(1);
```

Die untere Methode `Check()` wird von `_root.L1["Nuss"+this.ID].Nuss` in einer Schleife aufgerufen; dies muss unterbunden werden, da die Nuss gefangen wurde.

```
   _root.L1["Nuss"+this.ID].Nuss.gotoAndStop(Nope);
}
```

Diese Methode wird von jeder zu Boden fallenden Nuss in einer Schleife aufgerufen. Jede Nuss prüft, ob sie gefangen wurde oder nicht. Die Methode implementiert also die Fangroutine.

```
function _Nuss_Check_ () {
```

Nur wenn das Fangen nicht gesperrt ist, dann …

```
   if (_root.myKnilch.CatchAvail) {
```

Wenn eine Nuss den Netzbereich (`hitarea1`; siehe »Erstellung des Knilchs«) berührt, dann wird versucht, die Nuss im Netz aufzunehmen; siehe Methode `AddPoints();`.

```
      if (_root.L1["Nuss"+this.ID].hitTest
(_root.Player.KnilchAnim.Anim.Netz.hitarea1)) {
         this.AddPoints();
```

Wenn eine Nuss den Bereich `hitarea2` (siehe »Erstellung des Knilchs«) berührt, und der Knilch die fallende Nuss in sein Netz aufnehmen kann, dann soll der Knilch sich automatisch kurz umdrehen und die Nuss fangen.

```
      } else if (_root.L1["Nuss"+this.ID].
hitTes(_root.Player.KnilchAnim.Anim.Netz.hitarea2)
               && _root.myKnilch.NetzNutCount<3) {
         if (_root.myKnilch.KnilchAvail) {
            if (_root.myKnilch.LR == "Left") {
               _root.myKnilch.LR = "Right";
```

```
            } else {
               _root.myKnilch.LR = "Left";
            }
            _root.Player.KnilchAnim.gotoAndStop("Turn");
         }
         this.AddPoints();
      }
   }
}
```

Die obigen Funktionen werden der Prototype-Eigenschaft des Objekts **ObjNuss** zugewiesen und werden somit zu dessen Methoden.

```
ObjNuss.prototype.AddPoints = _Nuss_AddPoints_;
ObjNuss.prototype.Check = _Nuss_Check_;
```

Methoden des ObjAffeGen-Objekts

Mit dieser Methode wird versucht, einen noch nicht aktiven Affen zu starten.

```
function _AffeGen_ActivateNewAffe_ () {
```

RandomAffeVal ist ein Wert <=10. Damit wird verhindert, dass diese Methode immer einen brauchbaren Wert zwischen 1 und 12 errechnet und mit **AffenActiveMax** = 3 Affen gleichzeitig aktiv werden.

```
   this.RAffe = random(this.RandomAffeVal)+1;
```

Wenn **RAffe <= AffenCount (12)** ist, die Anzahl der aktiven Affen nicht das erlaubte Maximum überschreitet und der zu startende Affe nicht bereits aktiv ist, dann starte den Affen mit der **ID = RAffe (= 1 bis 12).**

```
   if (this.RAffe<this.AffenCount+1 && this.AffenActive-
Count<this.AffenActiveMax && !this.AffeAvail[this.RAffe]) {
      _root.L1["Affe"+this.RAffe].gotoAndPlay("Start");
   }
}
```

Wenn eine Nuss zu Boden fällt oder gefangen wird, kann der Affe, der diese Nuss geworfen hatte, wieder zum Start freigegeben werden. Außerdem muss die Anzahl der aktiven Affen aktualisiert werden.

```
function _AffeGen_DecAffe_ (ID) {
```

Der Affe mit der richtigen ID wird zum Start freigegeben. Beispiel: Nuss1 hat die ID = 1 und wurde von Affe1 mit der ID = 1 geworfen. Wenn Nuss1 nun zu Boden fällt oder gefangen wird, wird diese Methode von Nuss1 oder einer der Methoden von `myNuss1`-Objekt aufgerufen. `DecAffe` (ID) wird also mit der ID der Nuss aufgerufen (ID = 1) und `AffeAvail` wird mit dieser ID = 1 angesprochen. Affe1 ist nicht mehr gesperrt.

```
   this.AffeAvail[ID] = false;
```

Verringere die Anzahl der aktiven Affen um eins.

```
   this.AffenActiveCount--;
}
```

Sobald ein Affe gestartet wird, wird diese Methode von der geworfenen Nuss aufgerufen. Somit wird der Affe, der geworfen hat, für einen Neustart gesperrt, und die Anzahl aktiver Affen wird erhöht.

```
function _AffeGen_IncAffe_ (ID) {
```

Sperre Affe mit der richtigen ID.

```
   this.AffeAvail[ID] = true;
```

Inkrementiere Anzahl der aktiven Affen um eins.

```
   this.AffenActiveCount++;
}
```

Die obigen Funktionen werden der Prototype-Eigenschaft des `ObjAffeGen`-Objekts zugewiesen und werden somit zu dessen Methoden.

```
ObjAffeGen.prototype.ActivateNewAffe = _AffeGen_ActivateNewAffe_;
ObjAffeGen.prototype.DecAffe = _AffeGen_DecAffe_;
ObjAffeGen.prototype.IncAffe = _AffeGen_IncAffe_;
```

Methoden des ObjMobileGen-Objekts

Mit dieser Methode wird versucht, ein neues mobiles Fahrzeug zu starten. Zu jeder Zeit darf nur ein Fahrzeug aktiv sein.

```
function _MobileGen_ActivateNewMobile_ () {
```

Wenn kein Fahrzeug aktiv ist, dann ...

```
   if (!this.MobileInUse) {
```

Wichtig: sofortige Sperre, da nur ein Fahrzeug aktiv sein darf.

```
      this.MobileInUse = true;
```

Zufallswert; damit wird entschieden, ob das Fahrzeug von links oder rechts kommen soll.

```
      this.RandomMobile = random(2)+1;
      if (this.RandomMobile == 1) {
         this.LR = "Right";
      } else {
         this.LR = "Left";
      }
```

`MobileNum` läuft von 1 bis 4. Bei 1 startet der Truck, bei 2 das Flugzeug etc.

```
      this.MobileNum++;
      if (this.MobileNum>this.MobileForms) {
         this.MobileNum = 1;
      }
```

Welches Fahrzeug ist an der Reihe?

```
      if (this.MobileNum == 1) {
         _root.L1.PMobile1.gotoAndStop
(["Truck"+this.RandomMobile]);
      } else if (this.MobileNum == 2) {
         _root.L1.PMobile1.gotoAndStop
(["Flugzeug"+this.RandomMobile]);
```

```
      } else if (this.MobileNum == 3) {
         _root.L1.PMobile1.
gotoAndStop(["Cabrio"+this.RandomMobile]);
      } else if (this.MobileNum == 4) {
         _root.L1.PMobile2.
gotoAndStop(["Boot"+this.RandomMobile]);
      }
   }
}
```

Die obige Funktion wird der Prototype-Eigenschaft des `ObjMobileGen`-Objekts zugewiesen und wird somit zu dessen Methode.

```
ObjMobileGen.prototype.ActivateNew
Mobile = _MobileGen_ActivateNewMobile_;
```

Erzeugung der Instanzen obiger Objekte

Jetzt haben wir alle benötigten Objekte implementiert. Die Implementierung erfolgt in einem Frame in der Hauptzeitleiste (nicht in einer Schleife).

Die nachfolgenden Instanzen habe ich im selben Frame erzeugt.

Zum Verständnis ein Vergleich mit MovieClips: Wir sind mit unseren selbst definierten Objekten auf demselben Stand wie fertige MovieClips in der Bibliothek. Um Instanzen von MovieClips aus der Bibliothek zu erzeugen, ziehen wir die Clips mit der Maus auf die Bühne. Bei selbst definierten unsichtbaren Objekten benutzen wir den `new`-Operator und die Konstruktorfunktion des Objekts.

Instanz von `ObjScroll`; gescrollt wird mit zehn Pixeln. Spielebene L2 scrollt mit 7 = 10 – 3 Pixel. Spielebene L3 scrollt mit 3 = 10 – 7 Pixel. 484 ist der äußerste linke Rand des Spielfeldes und –8 der äußerste rechte Rand.

```
myScroll = new ObjScroll(10, 3, 7, 484, -8);
```

Instanz von `ObjKnilch`.

```
myKnilch = new ObjKnilch();
```

Instanzen der Fallen: ein Krebs, ein Loch und zwei Spinnen. `mySpinne1` wird mit der ID = 1 erzeugt und `mySpinne2` mit ID = 2.

```
myKrebs = new ObjKrebs();
myLoch = new ObjLoch();
mySpinne1 = new ObjSpinne(1);
mySpinne2 = new ObjSpinne(2);
```

Instanzen für Zeit und Score.

```
myTime = new ObjTime(2, 0);
myScore = new ObjScore();
```

Wir haben 22 Nüsse auf der Bühne. Daher erzeugen wir 22 Instanzen dieses Objekts jeweils mit der ID 1 bis 22.

- Die Nüsse 1 bis 12 werden von den Affen in den Palmen und Felsen geworfen. Diese Nüsse erzielen jeweils fünf Punkte.
- Nüsse 13 bis 16 werden von dem Flugzeug geworfen. 25 Punkte.
- Nüsse 17 und 18 werden vom Cabrio und Truck geworfen. 20 Punkte.
- Nüsse 19 bis 22 werden vom Boot geworfen. 10 Punkte.

```
myNuss1 = new ObjNuss(1, 5);
myNuss2 = new ObjNuss(2, 5);
myNuss3 = new ObjNuss(3, 5);
myNuss4 = new ObjNuss(4, 5);
myNuss5 = new ObjNuss(5, 5);
myNuss6 = new ObjNuss(6, 5);
myNuss7 = new ObjNuss(7, 5);
myNuss8 = new ObjNuss(8, 5);
myNuss9 = new ObjNuss(9, 5);
myNuss10 = new ObjNuss(10, 5);
myNuss11 = new ObjNuss(11, 5);
myNuss12 = new ObjNuss(12, 5);
myNuss13 = new ObjNuss(13, 25);
myNuss14 = new ObjNuss(14, 25);
myNuss15 = new ObjNuss(15, 25);
myNuss16 = new ObjNuss(16, 25);
myNuss17 = new ObjNuss(17, 20);
myNuss18 = new ObjNuss(18, 20);
myNuss19 = new ObjNuss(19, 10);
```

```
myNuss20 = new ObjNuss(20, 10);
myNuss21 = new ObjNuss(21, 10);
myNuss22 = new ObjNuss(22, 10);
```

Instanzen der zwei Spielfigurstarter-Objekte; die maximale Anzahl der aktiven Affen wird mit drei festgelegt. Zwölf Affen sind auf der Bühne vorhanden und vier verschiedene Fahrzeugtypen.

```
myAffeGen = new ObjAffeGen(100, 3, 12);
myMobileGen = new ObjMobileGen(4);
```

Erstellung der Controll MovieClips zum Aufruf der Methoden der Objekte

Jetzt definieren wir ein paar globale Funktionen. Mit deren Hilfe rufen wir die Methoden der Objekte auf. Diese globalen Objekte werden dann in Schleifen aufgerufen um das Spiel stets zu aktualisieren.

`GlobalControlStrg` ruft alle benötigten Methoden auf, um das Scrollen und die Bewegungen des Knilchs zu aktualisieren.

```
function GlobalControlStrg () {
   _root.myKnilch.SetMoveParams();
   _root.myKnilch.MoveKnilch();
   _root.myScroll.SetScroll();
}
```

`GlobalControlTrap` ruft alle benötigten Methoden der Fallen auf. Damit wird stets geprüft, ob der Knilch in eine Falle gegangen ist.

```
function GlobalControlTrap () {
   _root.myKrebs.Check();
   _root.myLoch.Check();
   _root.mySpinne1.Check();
   _root.mySpinne2.Check();
}
```

`GlobalActivateObjects` versucht stets Affen und Fahrzeuge zu starten.

```
function GlobalActivateObjects () {
```

```
    _root.myAffeGen.ActivateNewAffe();
    _root.myMobileGen.ActivateNewMobile();
}
```

Wir erstellen jetzt einen MovieClip mit zwei Frames.

Im ersten Frame:

```
_root.GlobalControlStrg ();
_root.GlobalActivateObjects ();
```

Im zweiten Frame:

```
_root.GlobalControlStrg ();
_root.GlobalControlTrap ();
gotoAndPlay (1);
```

Diesen MovieClip positionieren wir in der Hauptzeitleiste.

Ich habe mich entschieden, das Spiel mit 18 Frames pro Sekunde abzuspielen. Um die Spielzeit zu steuern, erstelle ich einen MovieClip mit 18 Frames.

Im ersten Frame:

```
_root.myTime.CountDown();
```

Im zweiten Frame:

```
_root.myTime.EndGame();
```

Die restlichen Frames bleiben leer.

Jedes Mal, wenn dieser Clip ein Mal durchläuft, wird die Spielzeit um eine Spielsekunde vermindert. Reale Zeit kommt für Flash-Spiele nicht in Frage, da man sonst auf langsameren Systemen keine Chance hat, in die Score-Liste zu kommen. Diesen MovieClip positionieren wir in der Hauptzeitleiste. Wir hoffen, dass Sie anhand dieses Beispiels einige Denkweisen nachvollziehen und sie in eigenen Spielen anwenden können. Nun gilt es, sich ein Konzept für ein eigenes Spiel zu machen und sehr strukturiert an die Programmierung zu gehen. Auch wenn Ihr Projekt nicht so komplex wird, wie wir es für die Traveltainment AG programmieren durften, sollten Sie versuchen, den Sourcecode möglichst übersichtlich zu gestalten. Funktionen und Objekte helfen dabei. Viel Spass!

Gestaltung

192 Grafik

192 Tipps zum Import von Pixelbildern
193 Blur-Effekt mit Flash
194 Übernahme von Screendesigns

198 Flash-Typografie

198 Warum Typografie?
199 Lesbarkeit
212 Basics

224 Animation mit Typografie

224 Texteffekte tweenen
225 Texteffekte als Skript
231 Der Schreibmaschineneffekt
233 Maskentexteffekt mit Duplicate Movie

236 Sound

236 Verfügbarkeit
237 Ein bisschen Theorie
239 MP3
240 Sounds aufnehmen und bearbeiten
247 Soundbearbeitung: Tipps
251 Musik selbst machen – MIDI & Co. – virtuelle und reale Studios
254 Alles inklusive (?)
256 SoundObject

264 3D

264 Probleme bei der Erstellung von 3D
265 Erstellungsmöglichkeiten
266 Mathematische 3D-Erstellung
266 3D mittels »Pseudos«
267 3D durch Sequenzen
269 Vom 3D-Modell zum Vektorbild
274 Vor- und Nachteile der einzelnen Tools
278 Workshop 3D-Clipping

Grafik

Effekte und Tipps für beste Grafik

Ein noch so guter Flasher ist ohne gute Grafik verloren. In diesem Kapitel möchten wir Ihnen ein paar Tipps zu Effekten, aber auch zu grundlegenden Problemen mit der Typografie oder der Screendesign-Übernahme geben.

Tipps zum Import von Pixelbildern

▸ Carlo Blatz

Flash kann diverse Formate importieren, aber nur wenige davon sind sinnvoll. Wir haben GIF, PNG, JPG und bei Windows BMP als mögliche Pixelbild-Importformate.

GIF verwendet man meistens, um Bilder zu importieren, die man auf wenige Farben reduzieren kann. PNG eignet sich dafür aber noch besser. Steht ein PNG in der Bibliothek auf Verlustlos, hat das Bild eine gestochen scharfe Qualität. PNG hat noch einen weiteren Vorteil: Man kann auch Alphaverläufe importieren, während man bei GIF nur eine Farbe auf transparent stellen kann. Allerdings steigt die Dateigröße bei transparenten Verläufen start an. Mit Flash 5 kann man nun auch aus Fireworks gespeicherte Vektoren und Texte im PNG-Format übertragen. Bitte beachten Sie: Wenn eine Fireworks PNG-Datei über die Zwischenablage importiert wird, wird die Datei in eine Bitmap konvertiert.

Neu in Flash 5 ist, dass man die Dateien in Fireworks PNG-Dateien auf eine Ebene reduzieren oder als editierbare Objekte importieren kann. Auf eine Ebene reduzierte Bilder werden mitsamt aller Vektorgrafiken gerastert oder in eine Bitmap konvertiert. Als editierbare Objekte importierte PNGs beinhalten auch nach dem Import in Flash alle Vektorinformationen. Man kann wählen, ob Text, Bitmaps und Führungslinien in der PNG-Datei behalten werden sollen.

Kann man ein Bild nicht auf zwei bis maximal 64 Farben reduzieren, eignet sich PNG nicht. Die meisten Flasher verwenden dann JPG. Allerdings lohnt es sich nicht, ein JPG vorher zu komprimieren und es dann in Flash erneut komprimieren zu lassen. Die Datei wird dadurch nicht klei-

ner, die Qualität nur schlechter. Ist JPG z. B. von Fireworks komprimiert, bietet Flash die Funktion Importierte JPG-Daten verwenden an.

Die besten Erfahrungen haben wir bei nicht farbreduzierbaren Bildern allerdings mit BMP gemacht. Gegen alle Gerüchte ist die Flash-interne JPG-Komprimierung nämlich gar nicht so schlecht, für beste Ergebnisse benötigt Flash allerdings ein unkomprimiertes Format wie BMP. In vielen Tests hat sich herausgestellt, dass importierte JPGs schlechter und größer sind als importierte BMPs, die Flash dann selbst komprimiert. Flash behält die Transparenzeinstellungen importierter Bitmaps übrigens bei. Dies funktioniert aber nur über die normale Importfunktion, in der Zwischenablage gehen die Transparenzeinstellungen verloren.

Die allerwichtigste – fast selbstverständliche – Regel ist allerdings, dass die Bilder mit 72 dpi (maximale Bildschirmauflösung) importiert werden und vor allem exakt mit der Zielgröße, mit der sie auf der Bühne dargestellt werden sollen. Es nützt nichts, ein Bild mit 600 x 600 Pixel zu importieren und es auf der Bühne auf nur 300 x 300 Pixel zu skalieren. Die Datei bleibt groß.

Wenn man eine Reihe von Pixelbildern hat – z. B. ein Video aus Einzelbildern, braucht man diese nicht einzeln von Hand zu importieren. Solange die Bilder als letztes Zeichen eine fortlaufende Nummer haben, erkennt Flash beim Import eines Bildes, dass es sich um eine Bildsequenz handeln könnte, und fragt, ob man die Bilder komplett importieren möchte oder nicht.

Blur-Effekt mit Flash

Es ist nicht einfach, mit Flash vernünftige Weichzeichnungseffekte zu kreieren. Will man z.B eine komplexe Schrift weichzeichnen, geht man meistens den Weg über Photoshop oder Fireworks und importiert das Bild.

Eine kleine Möglichkeit gibt es jedoch, auch in Flash Vektoren weichzuzeichnen. Sie verbirgt sich hinter dem Menü Modifizieren · Form · Ecken abrunden. Flash generiert aus den markierten Vektoren an allen Kanten Abstufungen bis zur Unsichtbarkeit. Das Programm fragt lediglich, in wie vielen Schritten es die Abstufungen erstellen soll und wie groß jeder Schritt in Pixel sein soll.

Zerlegt man z. B. eine Schrift und behandelt sie mit diesem Tool, sieht sie leicht verschwommen aus. Man kann das für Schatteneffekte nutzen.

Wenn man die eigentliche Form löscht, entsteht auch eine sehr interessante Outline.

Abbildung 1 ▶
Alle zwei Pixel eine neue Abstufung

Ist die Schrittweite zu groß gesetzt oder wird das Objekt zu groß skaliert, enttarnt sich der Effekt.

Vergleicht man die Dateigröße vor und nach der Eckabrundung, erhöht sie sich doch um einige KB. Flash generiert für jede Abstufung eine eigene Form und Füllung. Man sollte also sparsam damit umgehen. Je nach Menge der Anwendungen kann für diesen Zweck ein PNG oder JPG kleiner sein.

Ein Motionblur ist noch einfacher zu erstellen. Dafür stuft man das Bild oder den Vektor nur etwas ab und skaliert es bzw. ihn entsprechend in die gewünschte Richtung.

Einen Motionblur-Effekt kann man sich mit `Duplicate Movie` schnell selbst basteln. Ein Motionblur sollte so aussehen, als würde sich das Objekt schnell bewegen bzw. statisch so aussehen, als wäre ein Objekt bei einer schnellen Bewegung fotografiert worden. Der Effekt setzt sich in der Natur durch die Verzögerung des Lichts im Moment der Aufnahme zusammen. Diese Verzögerung können wir einfach nachbauen, indem das Objekt in Bewegungsrichtung jeweils eine Stufe transparenter wird.

Übernahme von Screendesigns

Seit Flash 5 ist klar, mit welchem Programm man die Daten vorbereiten sollte: FreeHand. Endlich werden die Daten eins zu eins importiert, sehen identisch aus und sind gleichwertig. Sogar Text bleibt als solcher endlich immer in einem Block. Vektoren sehen identisch aus, und Verläufe sind nach dem Import noch so, wie sie angelegt wurden. Letzteres funktionierte auch schon in Flash 4 über den SWF-Export – Texte hingegen bestanden aus diversen Textfeldern (was unhandlich ist und die Dateigröße unnötig erhöht). Über den Import von AI-Files (Adobe Illustrator) waren die Texte zwar brauchbarer, dafür aber die Verläufe abgestuft.

◂ **Abbildung 2**
Leicht verschwommen

◂ **Abbildung 3**
Ein an dreidimensional anmutendes Relief

◂ **Abbildung 4**
Stark vergrößert enttarnt sich der Effekt.

Wir konzentrieren uns aber auf die Übernahme aus FreeHand. So gut es seit Flash 5 funktioniert, gibt es doch noch einige Dinge zu beachten.

Die wenigsten Probleme entstehen, wenn man von vornherein in FreeHand mit Ebenen arbeitet und beachtet, dass bei Linien eckige Kanten automatisch in runde Kanten umgewandelt werden. Bedenken sollte man auch, dass platzierte Graustufenbilder in RGB-Bilder umgewandelt werden. Wenn Objekte in Flash nach dem Import aus FreeHand nicht mehr zu sehen sind, handelt es sich vermutlich um nach FreeHand importierte EPS. Um EPS in FreeHand zu importieren und diese in Flash zu erhalten, muss in den Importeinstellungen von FreeHand Editierbare EPS beim Importieren konvertieren aktiviert sein.

Flash kann aus FreeHand Farbverläufe mit maximal acht Farben übernehmen. Verwendet man mehr als acht Farben in einem Verlauf, improvisiert Flash mit gestuften Farbverlaufsunterteilungen, und die Datei wird größer. Wenn man in FreeHand mit Effekten arbeitet, importiert Flash diese Stufen als einzelne Frames, aber nicht mit dem Farbeffekt eines Symbols, sondern als jeweils neue Pfade. Die Datei wird hierbei ebenfalls unnötig größer.

In der Praxis gibt es aber noch mehr Probleme – **Pixelbilder** werden meist nicht mit übernommen, und wenn, dann nur in einer schlechten Qualität. Man sollte sie sich also extern (BMP in 72 dpi in Endgröße) übergeben lassen und von Hand einbauen. Sofern sie innen eingefügt sind,

existiert die Maske dafür ja auch bereits. Diese Option funktioniert übrigens hervorragend. Auch vier Mal ineinander eingefügte Objekte sehen noch wie gewünscht aus. Lediglich die Symbole, die Flash dafür erstellt, könnte man der Gründlichkeit halber auf Filmsequenz umdefinieren.

Kleine Mankos gibt es noch beim Text. Die **Laufweite** (Spacing) wird noch nicht richtig übertragen. Sofern ein Blocksatz angegeben wurde, vergrößert er lediglich den Abstand zwischen den Wörtern. Auch bei unterschiedlichen Zeilenabständen in einem Textfeld tut er sich schwer.

Text auf und in Pfaden (FreeHand-Funktion: TEXT MIT PFAD VERBINDEN) übernimmt Flash leider auch nicht – diese Texte müssten vorher in Vektoren umgewandelt werden.

Um etwas Handarbeit kommt man demnach nicht herum, mit wenigen Handgriffen kann man das Screendesign aber relativ exakt übernehmen.

Flash-Typografie

Wie gehe ich mit Schriften richtig um?

In diesem Kapitel beantworten wir Ihnen einige grundsätzliche Fragen zu Schrift und Typografie und vermitteln Ihnen einen kleinen Überblick auf diesen Themenbereich. Dabei ist vor allem interessant, wie Schrift auf dem Monitor in einem Flash-Film wirkt und was man als Gestalter deswegen beachten muss.

Warum Typografie?

▸ Michael Bundscherer
www.typolis.de

Es gibt viele Möglichkeiten, am Abend essen oder trinken zu gehen. Sie können in eine Bar gehen, in ein gutbürgerliches Wirtshaus, in eine Weinstube, in ein Fast-Food-Restaurant oder auch in eine In-Kneipe. Man kann Bier trinken, Cocktails, Cola oder Kaffee; Pizza essen, Fisch, Gyros oder Dampfnudeln. Aber wenn Sie sich zu einem Date verabreden, dann gibt es schon weniger Möglichkeiten. Laden Sie Ihre Flamme in ein normales Wirtshaus ein, dann hält sie/er Sie vielleicht für spießig, in der In-Kneipe ist es womöglich so laut, dass der Abend dort nicht lange dauern wird. Vielleicht ist sie/er auch Vegetarier/in? Sie werden sich deshalb sicher Gedanken machen, wo es Ihnen beiden gefällt und wo man gut miteinander reden kann.

Was hat das aber nun mit Flash zu tun? Nun, hier ist es genauso! Wenn Sie jemanden einladen, auf Ihre Seite zu kommen, dann sollten Sie dafür sorgen, dass sie/er sich wohl fühlt und die Informationen, die Sie vermitteln wollen, auch ohne Probleme aufnehmen kann. Erreichen können Sie das durch gute Typografie!

Typografie ist die als Kunst oder Handwerk verstandene Gestaltung mittels Schrift, Flächen und Räumen, Bildern, Linien und Farben. Typografie ist die **Inszenierung** einer Mitteilung! Wie das dann aber genau gemacht werden muss, dafür gibt es kein Patentrezept. Schließlich gibt es – um noch einmal auf das obige Beispiel zurückzukommen – mindestens genauso viele Möglichkeiten, den Abend zu verbringen, wie es Leute

gibt, mit denen Sie dort hingehen können. **Different strokes for different folks** – jedem das Seine.

Wenn Sie sich mit der Aufbereitung von Texten genauer beschäftigen, wird Ihnen auffallen, dass es prinzipiell zwei Möglichkeiten gibt: Man kann Texte besonders gut lesbar machen (klassische Typografie) oder Texte auffällig gestalten (experimentelle Typografie). Klassische und experimentelle Typografie stehen aber nicht im Widerspruch zueinander. Werden zum Beispiel plakative Schriften maßvoll eingesetzt, kann das natürlich auch ein Anreiz sein, sich anschließend mit längeren Texten auseinander zu setzen. Auch gute klassische Typografie kann schön aufbereitet sein. Seien Sie also experimentell, wo es Sinn ergibt, und achten Sie auf gute Lesbarkeit, wo es nötig ist!

Die Notwendigkeit, dass man Informationen lesegerecht aufbereiten muss, gab es natürlich schon vor Flash – genauer gesagt, seit Gutenberg im 15. Jahrhundert die beweglichen Bleilettern erfunden hat und man somit das Aussehen einer Publikation von vornherein reproduzierbar festlegen konnte. In der Zeit bis heute hat man auf Papier und viel später auch auf dem Monitor vieles ausprobiert. Manches hat sich bewährt, anderes eben nicht. Im folgenden Kapitel möchte ich Ihnen etwas von Typografie erzählen, damit Sie wissen, worauf Sie achten können.

Bevor es jetzt nun aber wirklich losgeht, noch einmal ein Vergleich aus der kulinarischen Ecke: In jedem Kochbuch finden sich viele gute Rezepte. Das heißt aber nicht, dass der beschriebene Weg der einzige ist, um ein gutes Gericht zu kochen. Was zählt ist das, was am Schluss auf dem Tisch steht!

Lesbarkeit

Flash ist für größere Textmengen im Web nicht unbedingt das am besten geeignete Tool. Aber da schließlich auch auf Flash-Seiten Informationen übermittelt werden müssen, kommen wir um grundsätzliche Überlegungen zur Lesbarkeit nicht herum.

Wenn wir von **Erkennbarkeit** der Schriften auf dem Monitor sprechen, meinen wir zunächst, wie eindeutig man die einzelnen Wörter und die darin enthaltenen Zeichen unterscheiden kann. Eine gute **Lesbarkeit** aber können wir dann erreichen, wenn einzelne Buchstaben nicht nur zu erkennen, sondern auch ganze Texte ohne Anstrengung zu lesen und nicht zu buchstabieren sind.

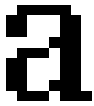


Abbildung 1 ▲
Ein kleines a bei einer Monitorauflösung von 72 dpi

Abbildung 2 ▲
Der Flash-Player zeichnet die Treppenstufen weich

Monitore können mit ihrer Auflösung von 72 dpi Details nur viel schlechter darstellen, als es z. B. beim Druck (2540 dpi und höher) möglich ist. Weil wir, um Schriften angenehm lesen zu können, auf Details angewiesen sind, fällt das besonders unangenehm auf. Die Buchstaben zeigen wegen der geringen Auflösung zum Beispiel an Rundungen »Treppenstufen« (wie etwa Schriften auf normalen HTML-Seiten). Die Zeichen müssen ja irgendwie in das Pixelraster passen.

Der Flash-Player hilft sich deshalb mit einer Methode, die Antialiasing genannt wird. Dabei werden problematische Stellen leicht unscharf angezeigt. Eine solche »weichgezeichnete« Schrift ist nicht mehr an das Raster gebunden; der Betrachter empfindet sie als schärfer. Das Problem hierbei ist aber, dass dadurch wichtige Details der einzelnen Zeichen verschwinden bzw. verschwimmen, die für die Lesbarkeit aber sehr wichtig sind. Besonders auffällig sieht man das bei kleineren Schriftgrößen. Verwendet man größere Schriften, dann verbessert sich das Schriftbild durch Antialiasing nämlich erheblich.

In den folgenden Bildschirmabzügen habe ich ein paar Schriften exemplarisch in verschiedenen Größen abgesetzt. Das (Phantasie-)Wort »Illumafontrn« soll uns helfen, Problemstellen einer Schrift auf dem Monitor zu erkennen: Das große I und das kleine l (wie am Anfang meines Beispielwortes) sehen bei vielen Schriften fast gleich aus (sinnigerweise auch hier in diesem Buch). Und wenn ein r und ein n zu dicht aneinander stehen, erkennen wir häufig nur ein m. Außerdem habe ich am Beispiel der Times New Roman ausprobiert, wie Modifikationen sich auf die Lesbarkeit einer Schrift auswirken.

Wenn ich in der Beschreibung zu den einzelnen Schriften Größenangaben mache, dann stellen diese Werte eine Mindestgröße dar. Man tut gut daran, diese Schwellwerte nicht zu sehr auszureizen. Ein paar Pixel mehr, und die Lesbarkeit nimmt weiter zu. Kann der Film später im Browser des Betrachters skaliert werden, muss man das ebenfalls bei der Wahl der Schriftgröße vorher beachten.

Die Abbildungen erhalten Schriftbeispiele stets in den Größen 6 px, 7 px bis 12 px, 14 px, 16 px, 18 px, 20 px.

Schriften

- **Arial**
  Die Arial wurde 1982 von Monotype (Robin Nicholas und Patricia Saunders) für das Betriebssystem Windows erstellt. Diese Schrift sollte sowohl auf dem Monitor als auch auf Desktop-Druckern gut aussehen. Als Vorlage diente hier die erfolgreiche Helvetica. Die Tabelle beweist auch, dass diese Schrift für den Bildschirm gut geeignet ist. Problematisch sind nur die ersten drei Zeichen des Beispielwortes in der Tabelle, die sich kaum unterscheiden. Die anderen Zeichen kann man ab 10 bis 11 Pixel erkennen. Lesbar wird diese Schrift aber erst ab 12 bis 14 Pixel und größer.

- **Times New Roman**
  Microsoft beauftragte Monotype auch, eine Schrift mit Serifen zu erstellen. 1987 konnten Ronald Carpenter und Robin Nicholas die Times New Roman vorzeigen. Sie ist eine Neuzeichnung der Times von Stanley Morison aus dem Jahr 1932, die für die gleichnamige Zeitung entwickelt wurde. Hier können wir deutlich das Manko einer Serifenschrift in Flash erkennen. Die feinen Serifen sind in kleineren Größen überhaupt nicht zu erkennen, selbst in größeren Graden werden sie noch stark weichgezeichnet. Auch das r und n lösen sich erst ab 12 Pixel voneinander. Damit die Times New Roman lesbar wird, muss sie etwas größer gesetzt werden als die Arial (ab 14 bis 16 Pixel).

- **Bodoni**

  Schauen wir uns doch Serifenschriften noch etwas genauer an. Spontan fällt mir hier zum Beispiel noch die Bodoni ein, die der geniale Italiener Giambattista Bodoni in der Blüte des Klassizismus 1790 erstellt hat (wir haben hier natürlich nur eine Nachzeichnung). Das Schöne an dieser Schrift sind die feinen Serifen und das sehr ausgewogene, fast strenge Schriftbild. Aber genau das ist es, was in diesem Beispiel gar nicht wirkt. Die feinen Striche und Serifen kann man sogar bei den größeren Zeichen nur erahnen. So ist zum Beispiel das kleine a erst ab 16 bis 18 Pixel als solches gut zu erkennen. Lesbar ist die Bodoni deshalb aber noch nicht. Ich empfehle sie für den Monitor ab einer Mindestgröße von 20 Pixel.

- **Garamond**

  Ein weiterer bekannter Schriftkünstler ist der Franzose Claude Garamond (15. Jahrhundert). Eine Interpretation seiner Renaissanceschrift habe ich hier abgebildet. Eigentlich gilt im Printbereich diese Schrift als besonders gut lesbar. Betrachtet man sie aber im Flash-Player, hat sie ähnliche Probleme wie die Bodoni, allerdings nicht ganz so ausgeprägt. Trotz des etwas zarten Schriftkörpers kann man die Buchstaben schon ab etwa 12 Pixel erkennen. Die Garamond ist aber erst ab 18 Pixel lesbar.

- **Centennial**

  Die Centennial ist eine etwas stabilere Schrift. Die Serifen sind ein wenig ausgeprägter, und die Zeichen kommen etwas stabiler daher. Adrian Frutiger hat sie 1986 entworfen. Allerdings kann man die einzelnen Zeichen ebenfalls erst ab 12 bis 14 Pixel erkennen und Texte ab 16 bis 18 Pixel einigermaßen angenehm lesen.

- **Egyptienne**

  Da die Serifen scheinbar ein Problem sind, habe ich jetzt eine Schrift ausprobiert, die sehr starke Serifen besitzt: eine so genannte serifenbetonte Linear-Antiqua. Die Egyptienne (1956) stammt ebenfalls aus der Hand von Adrian Frutiger. Frutiger interessiert sich schon lange für die Optimierung der Lesbarkeit von Schriften auf verschiedenen Ausgabemedien und für verschiedene Wahrnehmungsumfelder. Von ihm stammen nicht nur Schriften für den Bleisatz, Fotosatz und Computersatz, sondern auch für die Leit- und Informationssysteme verschiedener Flughäfen, der Pariser Metro und der Olympischen Spiele in München. Auch die OCR-B, eine der frühen computerlesbaren Schriften, stammt von ihm. Aber wieder zurück zur Egyptienne. Sie wirkt zwar wegen der stabilen Formen und Serifen etwas plump, ist aber gerade deshalb im Flash-Player für eine normale Serifenschrift gut lesbar. Positiv ist auch die unterschiedliche Ausgestaltung des großen I und des kleinen l. Obwohl nicht für den Monitor erstellt, ist sie aus diesen Gründen besser lesbar als die Times New Roman. Man erkennt die Buchstaben ab etwa 12 Pixel und kann sie ab 14 bis 16 Pixel relativ gut lesen.

- **Verdana**

  Die Verdana ist eine serifenlose Schrift, auch wenn sie zur besseren Erkennbarkeit beim großen I oben und unten Abschlussstriche besitzt. Das macht sie unter anderem am Bildschirm gut lesbar, was sicher auch die Absicht von Microsoft war, als man diese Schrift bei Matthew Carter in Auftrag gegeben hat. Positiv an dieser Schrift sind außerdem die relativ großen Innenräume (Punzen) zum Beispiel im kleinen a oder o. Dadurch werden die Buchstaben schon ab 9 Pixel erkennbar. Die Schrift sollte aber nicht unter 12 Pixel verwendet werden, wenn sie auch gelesen werden soll.

- **Futura**

  Ein paar Jahrzehnte hat die Futura schon auf dem Rücken. Sie wurde in den Dreißigerjahren des 20. Jahrhunderts von Paul Renner erstellt. Sie wirkt konstruiert und sachlich, hat aber entgegen anderen vergleichbaren Schriften einen eigenen Charakter. Eine Besonderheit dieser Schrift ist das runde a, das aber bei einer Schriftgröße unter 11 bis 12 Pixel auf dem Monitor leicht mit einem o verwechselt werden kann. Entscheidet man sich für diese Schrift, sollte man sie deshalb nicht unter 16 Pixel verwenden.

- **Avenir**

  Eine ähnliche Anmutung wie bei der Futura wollte Adrian Frutiger bei dieser 1988 entstandenen Schrift erreichen. Nicht nur der Name der Avenir (Zukunft) ist an die Futura und deren Stil angelehnt. Allerdings mutet bei ihr das Schriftbild nicht ganz so streng an. Das wirkt sich auch auf dem Bildschirm durch eine bessere Erkennbarkeit der Zeichen aus (in meinem Beispiel bei 11 Pixel). Lesen kann man den hier gezeigten Schnitt Nummer 55 bereits ab 14 Pixel.

- **Frutiger**

  Noch einmal eine Schrift von Adrian Frutiger, die diesmal den Namen ihres Entwerfers trägt. Wie schon erwähnt, wirken die vorhergehenden serifenlosen Schriften eher konstruiert und entsprechen somit dem Formgefühl von Anfang bis Mitte des 20. Jahrhunderts. Die Frutiger hingegen war, als sie 1976 erschien, etwas ganz Neues. Adrian Frutiger erkannte, dass die offeneren Formen (vor allem beim a, c und e) die einzelnen Zeichen unverwechselbar machen und damit die viel beschworene Lesbarkeit verbessern. Die Proportionen der Frutiger ähneln eher der einer Renaissanceschrift, nur eben ohne Serifen. Dass Renaissanceschriften gut lesbar sind, haben wir bei der Garamond schon bemerkt. Nur die Serifen und der zarte Schriftkörper störten hier. Bei der Frutiger kann man die Zeichen bei 10 bis 11 Pixel erkennen und Texte ab ca. 14 Pixel lesen.

- **Gill Sans**

  Der Engländer Eric Gill hat sich Anfang des 20. Jahrhunderts einen Namen als Bildhauer gemacht. Diese Arbeit brachte es mit sich, dass er sich auch mit (In-)Schriften auf Grabplatten usw. beschäftigen musste. Im Laufe seiner umfangreichen Tätigkeiten erstellte er auch ein paar Schriften, zum Beispiel die in der Tabelle abgebildete Gill Sans. Diese 1929 entstandene Schrift hat, wie schon die Frutiger, den Charakter einer Renaissanceschrift. Weil aber die einzelnen Zeichen wie bei dieser nicht so offen sind, ist auch die Erkennbarkeit etwas schlechter (ab 12 Pixel). Zu Irritationen kann auch die etwas ungewöhnliche Form der Ziffer 1 führen, die nur aus einem senkrechten Strich besteht. Lesbar ist sie ab 14 bis 16 Pixel.

- **Syntax**
  Die Syntax, die letzte hier gezeigte Serifenlose, stammt von Hans Eduard Meier aus dem Jahr 1968. Im Prinzip gilt hier dasselbe wie schon bei der Gill, nur dass die Buchstabenformen hier ein etwas eigenwilligeres Aussehen haben. Das hat zur Folge, dass die Zeichen schon ab 9 bis 10 Pixel zu erkennen sind, die Schrift selbst aber ebenfalls ab 14 bis 16 Pixel lesbar ist.
  Dies ist übrigens die Schrift, in der dieses Buch gesetzt ist.

- **Kuenstler Script**
  Dass Effektschriften nicht für längere Lesetexte geeignet sind, dürfte jetzt eigentlich schon klar sein; das führt uns die Kuenstler Script noch einmal beispielhaft vor Augen. Hier dürfte ein unbedarfter Leser auch noch bei 20 Pixel Probleme haben, die Buchstaben zu entziffern. Ist am Anfang meines Wortes ein H oder doch ein J? Man könnte diese Schrift durchaus in Flash verwenden, dann aber bitte genügend groß und nur für einzelne Wörter, die visuell in diesem Stil umgesetzt werden können.

- **HiScore**

  Dann gibt es noch Schriften, die Einschränkungen des Pixelrasters auf dem Monitor als Stil bewusst betonen. Die HiScore zum Beispiel basiert selbst auf einem groben Rastersystem, das in der Höhe neun Rastereinheiten besitzt. Deshalb muss der Flash-Player bei einer Schriftgröße von genau dieser Größe auch kein Antialiasing einsetzen. Die Schrift ist bei 9 Pixel (und dem Vielfachen) in der Tabelle gestochen scharf. Möchte man sich nicht an diese Größen halten, muss die Schrift aber sehr groß dargestellt werden, um lesbar zu sein. Leider sind solche Schriften aber wegen ihrem eigenwilligen, technohaften Charakter nur dort einsetzbar, wo auch diese Wirkung erwünscht ist. Weil die Buchstaben anders geformt sind, als der Betrachter es von Leseschriften gewohnt ist, eignet sich diese Schrift auch nur für Stichpunkte (zum Beispiel in der Navigation).

- **Hi WebT**

  Noch ein Beispiel einer Pixelschrift. Es gilt hier dasselbe wie bei der vorhergehenden Schrift: scharf bei 9 und 18 Pixel. Allerdings ist die Hi WebT eine ziemlich fette Schrift, die noch dazu nur aus Großbuchstaben besteht. Die einzelnen Buchstaben sind nicht nur bei 9 Pixel nicht genügend gut erkennbar und unterscheidbar. Die Erkenntnis, die wir daraus ziehen können: Wenn eine Schrift einfach nur scharf dargestellt wird, bedeutet das noch lange keine gute Lesbarkeit.

- **sans**

  Man kann Flash auch anweisen, bei einem (statischen) Text die Schriftkonturen nicht einzubinden. Wenn man als Schriftart _sans einstellt, wird der Text beim Betrachter lediglich in einer serifenlosen Standardschrift angezeigt, je nach System meist Arial oder Helvetica. Der Vorteil: Die SWF-Datei bleibt klein, und die Schrift wird nicht antialiased, sondern gepixelt angezeigt. Somit wird die auf meinem Computer verwendete Arial bereits ab 10 Pixel relativ gut erkennbar und lesbar. Es gibt hier aber auch Nachteile. Das Schriftbild schaut (vor allem in größeren Graden) nicht mehr besonders ästhetisch aus. Modifikationen, wie zum Beispiel die Erhöhung der Laufweite (Sperren) oder Drehen des Textes, sind nicht möglich.

- **serif**

  Stellt man als Schriftart eine _serif ein, wird zum Beispiel die Times New Roman angezeigt. Auch sie wird nicht weichgezeichnet und ist deshalb pixelig und scharf, deutlich bemerkt man das bei den Serifen. Hier sind aber leider die einzelnen Buchstaben zu dicht zusammen, so dass sie an einigen Stellen eine Verbindung miteinander eingehen. Das r und n könnte auch bei 12 bis 14 Pixel noch als m gelesen werden. Die Serifen sind auch hier in kleinen Größen eher hinderlich.

Modifikationen

Nicht nur die verwendete Schrift beeinflusst die Lesbarkeit, sondern auch die Art und Weise, wie man mit ihr umgeht. In den nächsten Beispielen habe ich verschiedene Modifikationen an der Times New Roman ausprobiert.

- **Großbuchstaben**
  Wird ein Text in Versalien (Großbuchstaben) geschrieben, kann man die Zeichen schon in kleineren Schriftgrößen erkennen. Warum, das ist leicht nachzuvollziehen: Großbuchstaben sind eben größer und wuchtiger und neigen nicht so leicht dazu, sich zu verbinden. Leider sind aneinander gereihte Versalien aber schlechter lesbar. Weiter unten in der Rubrik »Versalien« gehe ich näher darauf ein.

- **Bold**
  Kräftigere Schriften sind auf dem Monitor besser lesbar, das haben wir weiter oben schon festgestellt. Die einzelnen Buchstaben können sich gegenüber dem Hintergrund besser behaupten, werden von diesem nicht so leicht überstrahlt. Verwendet man aber nicht nur eine kräftigere Schrift, sondern eine fette Schrift (wie im Beispiel die Times New Roman bold), dann ist das eindeutig zu viel des Guten. Das kleine a ist in kleineren Größen nur noch ein dunkler Fleck, u und m scheinen nur noch aus vertikalen Strichen zu bestehen. Bei der Frutiger wurde schon beschrieben, dass für eine bessere Lesbarkeit die Buchstaben offen und die Innenräume gut erkennbar sein müssen. Bei diesem fetten Schnitt laufen sie aber eher zu. Auch zum Einsatz von fetten Schriften habe ich weiter unten bei »Auszeichnungsarten« noch etwas zu sagen.

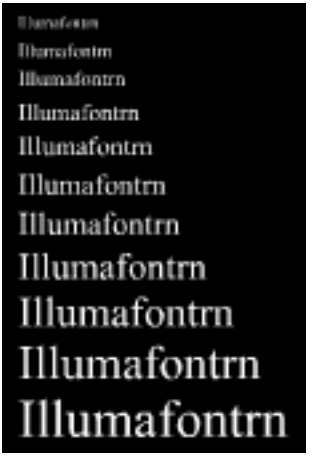
- **Kursiv**
  Fast jede seriöse Textschrift wird heute auch in einer kursiven Variante (Italic) angeboten. Im Gegensatz zur gerade stehenden Times New Roman ist die kursive aber sehr viel schlechter lesbar. Der Grund sind eben die Schrägen, die bei Antialiasing immer »weichgezeichnet« werden, was zu einem verschwommenen, unklaren Schriftbild führt.

- **Unterschnitten**
  Aus demselben Grund wie bei der fetten Schrift ist auch von einer Verringerung der Laufweite (Unterschneidung) abzuraten. Im Beispiel habe ich um einen Pixel unterschnitten, die Folgen für die Lesbarkeit sind aber fatal. Dass dies in Textgrößen auch nicht besonders ästhetisch ausschaut, stört dann schon gar nicht mehr

- **Gesperrt**
  Anders verhält es sich, wenn ich die Laufweite erhöhe (Sperrung). Die Schrift wird luftiger, die einzelnen Buchstaben werden im günstigsten Fall besser erkannt. Allerdings wäre (wie im Beispiel) ein Pixel schon ein viel zu hoher Wert. Das Wort ist nicht mehr als geschlossenes Ganzes erkennbar, sondern nur noch als Aneinanderreihung einzelner Buchstaben. Man muss also auch hier Vorsicht walten lassen. Bedenken Sie auch, dass der Schriftkünstler bei der Erstellung seiner Schrift meist Wochen oder Monate gebraucht hat, um den optimalen Abstand der einzelnen Buchstabenpaare zu ermitteln.

- **Aufgehellt**
  Wenn man darüber nachdenkt, verwundert es nicht, dass aufgehellte Schriften oder Schriften mit wenig Kontrast zum Hintergrund schlechter lesbar sind. Trotzdem gibt es hier zusätzliche Fallen. Als ich dieses Beispiel auf verschiedenen Monitoren betrachtete, konnte man auf dem einen die Schrift schon ab 14 bis 16 Pixel lesen, auf einem anderen Bildschirm noch nicht einmal in 20 Pixel Größe. Je nach Monitoreinstellung des Betrachters ist der Kontrast und die Helligkeit eines Flash-Films nicht unbedingt derselbe bzw. dieselbe wie beim Entwickler. Wenn man will, dass Text überall gelesen werden kann, sollte man mit der Schrift nicht zu risikofreudig umgehen. Ähnliche Probleme kann es übrigens auch geben, wenn der Kontrast der Schrift zu einem Hintergrundbild nicht groß genug ist!

- **Negativ**
  Die Times New Roman negativ, also helle Schrift auf dunklem Grund gesetzt, ergibt eine schlechtere Lesbarkeit. Der Hintergrund ist zu dominant und erdrückt die feinen Zeichen der Schrift. Das fällt wieder besonders bei den kleinen Größen auf, bei denen das Antialiasing deutlich spürbar ist. Wenn man auf eine negative Darstellung nicht verzichten kann/will, sollte man eine geeignete stabile und offene Schrift verwenden.



Zusammenfassung von Lesbarkeit

- Verwenden Sie große Schriften.
- Verwenden Sie Schriften mit klar unterscheidbaren und offenen Formen.
- Besser Schriften ohne Serifen.
- Schriften sollten keine zu dünnen Linien besitzen (aber auch nicht »bold«).
- Besser seriöse als verspielte Schriften.
- Achten Sie auf guten Kontrast zum Hintergrund.
- Verringern Sie nicht die Laufweite.

Basics

Zeilenbreite und Zeilenabstand

»**Einer ungeschickten Anordnung helfen selbst die allerschönsten Schriften nichts, aber selbst mit mittelmäßig guten Schriften kann man eine gefällige Anordnung treffen«,** sagte einmal der große »Papst der Typografie«, Jan Tschichold. Wir können uns also nicht allein darauf verlassen, dass die Buchstaben einer Schrift gut zu erkennen sind und die Schrift gefällt. Denn um einen Text ansprechend und gut lesbar zu gestalten, gehört natürlich noch mehr dazu.

Wir lesen Texte zeilenweise. Man muss am Ende einer Zeile mit den Augen in die nächste »springen« und dabei die ganze Textbreite überqueren. Logisch: Je weiter die Strecke ist, desto leichter verläuft sich das Auge, der Lesefluss wird gehemmt. Man muss also darauf achten, dass die Zeilen nicht zu lang sind.

Andererseits: Wenn die Zeilen zu kurz sind und zu wenige Wörter in eine Zeile passen, dann gibt es bei Blocksatz hässliche, ungleichmäßig große Löcher. Bei linksbündigem Flattersatz entsteht eine zu unruhige rechte Satzkante, weil die Differenz zwischen den längsten und den kürzesten Zeilen zu groß wird. Man nennt das dann eine zu große Flatterzone. Außerdem muss hier das Auge zu oft hin und her springen, was mit der Zeit durchaus ermüdend ist. Ideal befinden sich je nach verwendeter Schrift und Schriftgröße etwa 55 Anschläge oder mindestens sechs Wörter in den Zeilen. Dann ist der Text am günstigsten aufzubereiten und am besten lesbar.

Professionelle Gestalter erhöhen vor allem bei größeren Textmengen zugleich den Zeilenabstand. Damit wird das Wiederfinden der nächsten Zeile erleichtert.

Merkregeln

Vorsicht: Ist der Zeilenabstand zu groß, bewirkt das wieder das Gegenteil! In kleinen Schriftgrößen sind zusätzliche 2 bis 5 px geeignete Werte.

Je breiter die Zeilen, je geringer der Kontrast zum Hintergrund und je größer der Schriftgrad, desto größer muss der Zeilenabstand sein.

Zeilenabstand in Flash
In Flash kann man den Zeilenabstand in der Palette Absatz einstellen (Strg+⇧+T).

Zu kleine Ränder

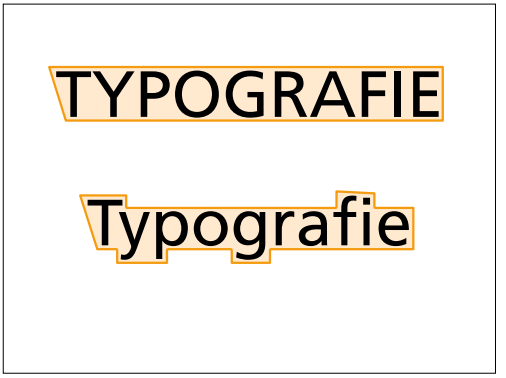
Noch ein Argument gegen zu lange Zeilen ist, dass so auf den Seiten zu wenig freier Raum bleibt. Auch Schrift will atmen! Je mehr Texte und grafische Elemente den Bildschirm bedecken, desto schwieriger wird es für den Betrachter, sich in diesem wichtigtuerischen Nebel effektiv zurechtzufinden. Deshalb: Längere Informationen lieber scrollbar machen oder auf mehrere Seiten verteilen. Denkbar wäre auch eine Lösung, in der man das »Yugop-Menü« von Seite 86 einsetzt.

Versalien

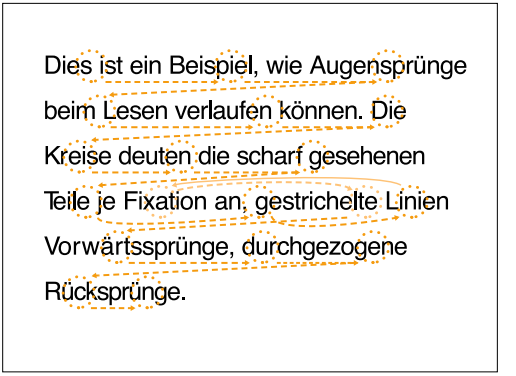
Und wenn wir schon einmal bei der Wahrnehmungspsychologie sind: Wie lesen wir eigentlich? Mal abgesehen von Leuten, die gerade eine Sprache neu erlernen, erfassen wir beim Lesen nicht die einzelnen Buchstaben. Das Auge springt schrittweise (in so genannten Sakkaden) über den Text und fixiert nur etwa 1/4 bis 1/3 Sekunden lang bestimmte Textstellen. Je schneller der Betrachter liest, desto größer werden die Sakkaden. In einer nicht zu großen Schrift werden aber maximal zwei bis vier Buchstaben scharf gesehen. Daraus folgt: Je besser die einzelnen Buchstaben zu unterscheiden sind, desto weniger genau muss man auch hinsehen. Anders gesagt: umso mehr kann man auch am Rande des scharfen Bereiches erkennen.

Wenn wir also ein Wort oder einen Satzteil lesen, erfassen wir nicht die einzelnen Buchstaben, sondern ganze »Wortbilder«. Schreiben wir ein Wort in Versalien, also in Großbuchstaben, dann sehen wir, grob gesagt, nur ein fleckiges Rechteck. Erst die Ober- und Unterlängen der kleinen Buchstaben machen ein Wortbild lebhaft und unterscheidbar. Versalien (Großbuchstaben) sind also schlechter lesbar und deshalb zu vermeiden oder zumindest nur sehr sparsam einzusetzen!

Manchmal kann es aber auch sinnvoll sein, einzelne Wörter in Versalien zu schreiben, zum Beispiel um etwas auszuzeichnen (hervorzuheben). Gut sortierte Schriftfonts besitzen für Versalien einen eigenen Schriftschnitt, die Kapitälchen (engl. Small Caps, häufig einfach Caps). Das sind einfache Großbuchstaben, die aber nur etwa so groß wie Kleinbuchstaben ohne Oberlänge sind. Somit stechen Kapitälchen im fortlau-



▲ Abbildung 3
Das Erste, was man von einem Wort erkennt, ist seine Silhouette. Wörter in Großbuchstaben werden deshalb schwerer erkannt.


Dies ist ein Beispiel, wie Augensprünge
beim Lesen verlaufen können. Die
Kreise deuten die scharf gesehenen
Teile je Fixation an, gestrichelte Linien
Vorwärtssprünge, durchgezogene
Rücksprünge.

▲ Abbildung 4
Beim Lesen wird nicht jeder Buchstabe einzeln betrachtet, sondern ganze Wort- und Satzteile. Großbuchstaben werden deshalb schwerer erkannt.

fenden Text zwar nicht mehr so stark hervor wie Versalien, passen sich dafür aber bedeutend harmonischer dem gesamten Schriftbild an. Versalien oder Kapitälchen sollte man leicht sperren, weil etwas Luft zwischen den Buchstaben die Wörter nicht so gepresst erscheinen lässt. Die normalen Buchstabenabstände sind schließlich für gemischten Text aus Groß- und Kleinbuchstaben vorgesehen und für reine Versalien und Kapitälchen zu eng.

Noch eine Anmerkung
Das ß ist ein Kleinbuchstabe! Setzt man ein Wort in Versalien oder Kapitälchen, wird es durch ein doppeltes s ersetzt. Also GROSSBUCHSTABE statt GROßBUCHSTABE.

Markierte Texte kann man in Flash in der Palette ZEICHEN sperren (Strg+T).

Auszeichnungsarten

Unter Auszeichnen versteht man die typografische Betonung einzelner Wörter oder Wortfolgen (z. B. Überschriften, Zitate, Aufzählungen, Links usw.). Typografische Mittel zur Auszeichnung können neben den bereits beschriebenen Versalien auch die Kursive, Kapitälchen, ein fetterer Schnitt oder eine andere Farbe der Schrift sein.

Ist ein kursiver Schnitt nicht vorhanden, verwenden viele Programme eine »unechte Kursive«, bei der die Buchstaben elektronisch schräg gestellt werden. Bei professionellen Gestaltern ist diese Möglichkeit aber verpönt, weil dadurch die Schrift »verhunzt« wird.

▲ Abbildung 5
Echte Kursive

Ist ein kursiver Schnitt nicht vorhanden, verwenden viele Programme eine »unechte Kursive«, bei der die Buchstaben elektronisch schräg gestellt werden. Bei professionellen Gestaltern ist diese Möglichkeit aber verpönt, weil dadurch die Schrift »verhunzt« wird.

▲ Abbildung 6
Unechte Kursive

Kursiv

Weil sie vom Leser am wenigsten störend empfunden wird, ist eine kursive Schrift (Italic) wohl die beste Auszeichnungsmöglichkeit. Im Idealfall nimmt er sie erst wahr, wenn er an die betreffende Stelle kommt. Kursiv bedeutet aber nicht einfach nur schräg. Tatsächlich ist das nur eine sekundäre Erscheinung! Die Kursive war zunächst eine eigenständige Schrift, die vermutlich im 15. Jahrhundert in florentinischen Schreibstuben entstanden ist. Weil die Schrift schnell geschrieben wurde (kursiv kommt vom lateinischen currere = laufen, eilen), kam es bei den meisten Schreibern zu einer mehr oder weniger deutlichen Rechtsneigung der Formen. Die gerade stehende Antiqua und die Kursive waren lange Zeit zwei verschiedene Schriften. Erst später wurde die Kursive zur Schwesternschrift der Antiqua, zur Auszeichnungsschrift. Das kann man noch an den zum Teil unterschiedlichen Formen des a und g erkennen. Deswegen reicht es nicht, eine Schrift einfach schräg zu stellen.

Bold

Manchmal kann es natürlich auch erwünscht sein, dass Auszeichnungen auffallen. Zum Beispiel um am Anfang eines neuen Abschnittes mit einer fetten Schrift dem Leser einen neuen Leseanreiz zu geben. Sie fällt schon auf, bevor der Leser mit dem Lesen des Textes beginnt. Häufig sieht man aber, dass hier maßlos übertrieben wird. Zu viele fette Auszeichnungen nehmen sich nämlich die Wirkung, machen den Text zu unruhig und schrecken so viele potenzielle Leser ab.

Farbe

Farbige Schrift als Auszeichnung muss einen ausreichend stabilen Schriftkörper haben. Die Farbwirkung würde sonst verloren gehen. Man könnte zum Beispiel eine fette Schrift zusätzlich einfärben, um somit zugleich den Helldunkelkontrast etwas abzuschwächen.

Andere Schrift

Kritisch ist auch der Einsatz einer völlig anderen Schriftart zur Auszeichnung. Man kann nicht jede beliebige Schrift verwenden. Sie muss einerseits stilistisch zur Grundschrift passen, sich aber andererseits deutlich von ihr unterscheiden. Je deutlicher der Kontrast zwischen beiden Schriftarten, desto leichter vermeidet man einen Konflikt zwischen beiden. Nicht verwandte und wenig unterscheidbare Schriftformen sollten allerdings nicht gemischt werden. Man kann diese Auszeichnungsart zusätzlich mit weiteren (andere Farbe, Kursive usw.) mischen. Eine zweite Schriftart als Auszeichnung zählt somit ebenfalls zu den Auszeichnungsarten, die der Leser sofort bewusst wahrnimmt. Ansonsten gilt hier dasselbe wie bei allen anderen auffälligen Auszeichnungsarten: Weniger ist oft mehr!

Und der Rest?

Andere Auszeichnungen wie Versalien, Sperrung, Unterstreichungen (ein Relikt aus der Schreibmaschinenära) oder eine größere Schrift sind lesehemmend, weil sie, zumindest in fortlaufenden Texten, meistens ein zu unruhiges Bild ergeben und somit den Leser vom eigentlichen Inhalt des Textes ablenken.

Links hervorheben

Links zu anderen Seiten sollten ebenfalls hervorgehoben sein, damit der Leser sie überhaupt findet. Überlegen Sie sich aber vorher, ob Sie wirklich eine Auszeichnung verwenden wollen, die gleich beim ersten Blick auf die Seite ins Auge springt. Oder anders gesagt: Will man dem Besucher beim Betreten einer Seite gleich zeigen, wie er möglichst schnell wieder hier wegkommt?

Welche und wie viele Schriften?

Eine ästhetisch schlechte Schriftwahl für die meisten Flash-Seiten ist meiner Meinung nach die Helvetica, die seit Beginn der 60er-Jahre wohl so etwas wie die Lieblingsschrift der Designer ist. Sie ist sachlich nüchtern

▲ Abbildung 7
Eine der weltweit am häufigsten eingesetzten Schriften ist die Helvetica.

▲ Abbildung 8
Jede Schrift hat ihre eigene Anmutung.

und passte so hervorragend zu der damals in Mode gekommenen »Schweizer Typografie«. Die Gestaltung war hier klar und funktional, Ornamente und Emotionen waren verpönt.

Im Laufe der Jahre führte dann aber die ständige Wiederholung und die damit einhergehende Beliebigkeit dieses Stils zu einem Verschleiß der Ausdruckskraft und zu einer Bürokratisierung. Trotzdem oder eben deshalb verwendeten viele große Unternehmen und öffentliche Einrichtungen bis zuletzt die Helvetica als ihre Hausschrift. Deshalb wirkt sie heute auch etwas uniformiert.

Da aber die Helvetica so häufig eingesetzt wurde, gab es natürlich auch viele Nachahmungen. Eine davon ist die Arial, die heute jeder Windows-Benutzer auf seinem Rechner findet und somit zur Standardschrift des Internet avancierte.

Ein großer Vorteil von Flash gegenüber HTML ist, dass man jede beliebige Schrift in einem Film verwenden kann. Benutzen Sie Ihre Fantasie, und geben Sie Ihrer Arbeit mit einer geeigneten Schrift Charakter, statt mit Standardschriften Beliebigkeit zu demonstrieren. Das Auge liest mit!

Schriften einbetten
Schriften werden beim Export in das SWF-Format automatisch eingebettet, wenn man in der Palette Textoptionen den Haken vor Schriftarten des Geräts verwenden weglässt (Statischer Text) oder bei Schriftarten einbetten die entsprechende Option auswählt (Dynamischer Text und Texteingabe).

Wenn H. P. Willberg sagt, »Schrift ist nicht einfach nur zum Lesen da, man sieht sie auch«, meint er damit auch, dass Schrift nicht nur Informationen transportiert, sondern auch Emotionen. Für einen eher wissenschaftlichen Text wird man keine Schreibschrift verwenden, für einen Liebesbrief keine Schrift, wie man sie auf Bauwagen findet. Jede Schrift hat ihren eigenen Charakter. Das trifft natürlich nicht nur auf Schmuckschriften zu, sondern auch auf ganz »normale« Leseschriften.

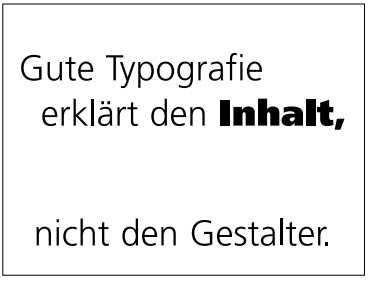
Tipps zur richtigen Schriftwahl:

- Suchen Sie sich je nach gewollter Aussage des Textes eine Schrift aus, die diese Aussage unterstützt.
- Drängen Sie sich nicht zwischen den Text und den Leser.
- Verwenden Sie gut lesbare Schriften (vor allem für den Grundtext).
- Systemschriften hat jeder und verwendet jeder. Verwenden Sie eine Schrift, die noch nicht »verbraucht« ist.

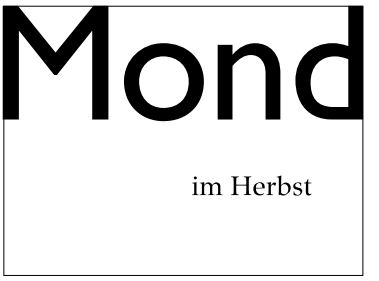
Wie man mit einer zweiten Schrift umgehen sollte, haben wir ja schon unter »Auszeichnungsarten« erfahren. Übertreiben Sie es aber nicht mit der Anzahl der verschiedenen Schriften. Anfänger machen oft den Fehler, dass sie in einem Flash-Film möglichst viele interessante Schriften unterbringen wollen – sozusagen als Trophäen ihrer Sammelleidenschaft. Zwei verschiedene Schriftarten reichen fast immer für eine Publikation! Hinzu kommt, dass jede neue verwendete Schrift die Dateigröße des Flash-Films vergrößert. Für normale Lesetexte verwendet man am besten eine gut lesbare, eher konservative Schrift, Überschriften und andere Blickfänge kann man dann dafür etwas experimenteller gestalten.

Kontraste

Es reicht aber nicht, die Lesbarkeit zu optimieren und den Text richtig aufzubereiten. Denn was nutzt Lesbarkeit, wenn nichts reizt, den Text überhaupt zur Kenntnis zu nehmen? Gute Typografie ist auch ein Ergeb-



▲ **Abbildung 9**
Fett-Fein-Kontrast



▲ **Abbildung 10**
Groß-Klein-Kontrast

nis wohl überlegter Anordnung von Texten, Bildern, Flächen usw. Gute Schrift, richtige Anordnung – das sind die beiden Säulen guter Typografie.

Wie kann man aber eine gefällige Anordnung oder gar Aufmerksamkeit erzeugen, ohne eine Seite wieder unruhig wirken zu lassen? Der Mitbegründer der »Neuen Typografie«, Paul Renner, würde uns etwas pathetisch darauf antworten: **»Die Aufgabe des Künstlers ist es, Ruhe und Unruhe so zu mischen, dass nicht erstarrte Unruhe entsteht, sondern jene gespannte Ruhe, die zum Ausdruck des Lebens wird.«**

Spielen Sie mit Kontrasten! Eine Einsatzmöglichkeit, wo man Kontraste verwenden kann, wurde ja schon angedeutet, als es um eine zweite Schriftart zur Auszeichnung ging. Es gibt aber natürlich noch viel mehr Möglichkeiten, zum Beispiel der Hell-Dunkel-, Groß-Klein-, Fett-Fein-, Farb-, Positiv-Negativ-, Form-, Und-so-weiter-Kontrast! Ich hoffe, der Appetit kommt beim Essen, die Inspiration aber kommt beim Arbeiten. Experimentieren ist gefragt!

Optische Achsen

Beachten Sie auch, dass Elemente, die nahe beieinander liegen (wie das auf dem Monitor der Fall ist), immer auch eine visuelle Verbindung miteinander eingehen. Schon die Urmenschen haben beim Betrachten des Sternenhimmels nicht einfach nur viele Punkte gesehen, sondern diese Punkte in Verbindung zueinander gebracht. Aus der Zusammenstellung der Sterngruppen sind Bilder entstanden – heute bekannt als Sternzeichen. Wenn wir also zum Beispiel zwei Textblöcke untereinander stellen oder neben einen Text ein Bild platzieren, dann gehen auch sie eine Ver-

▲ Abbildung 11
Hell-Dunkel-Kontrast

▲ Abbildung 12
Kontrast Künstlerisches – Sachliches

▲ Abbildung 13
Positiv-Negativ-Kontrast

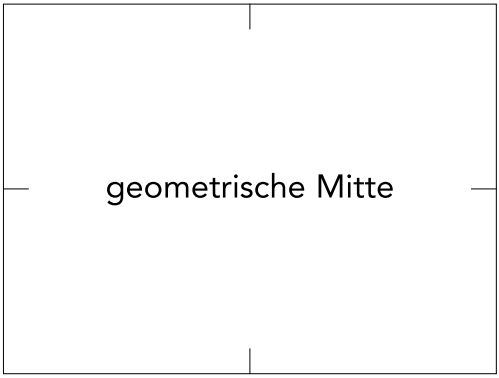
bindung ein. Wir nennen das dann eine optische Achse. Um eine klare Anordnung auf der Seite zu erreichen, sollten Sie darauf achten, dass so wenige optische Achsen wie möglich vorkommen.

Optische Mitte

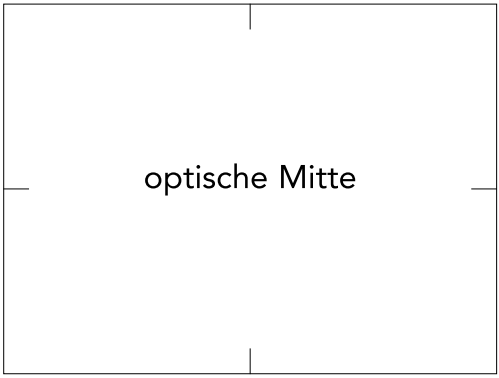
Wann sitzt ein Text genau in der Mitte meines Flash-Films? Klar, wenn der Raum darüber genauso groß ist wie der darunter. Die Abbildung zeigt aber, dass der Text bei dieser Anordnung scheinbar zu tief sitzt. Das menschliche Auge empfindet optisch die Mitte einer Fläche immer etwas höher als die tatsächliche, geometrische Mitte. Dieses Phänomen wird optische Mitte genannt.

Das ist auch ein Grund, warum bei Büchern der untere Rand meist größer ist als der obere. Auf dem Monitor kommen noch die Browserleisten oben hinzu. Die Mitte eines fensterfüllenden Films sitzt also sowieso schon etwas tief. Diese Tatsache muss bei der Gestaltung der Flächen und Proportionen eines Films berücksichtigt werden. Die optische Mitte liegt über der geometrischen.

Pop-up-Fenster platzieren
Auch JavaScript-Pop-up-Fenster sollten immer etwas über der mathematischen Mitte des Monitors platziert werden.



▲ Abbildung 14
Es gibt einen Unterschied zwischen geometrischer ...



▲ Abbildung 15
... und optischer Mitte.

Die vielen Effekte

Manchmal kann man auf Flash-Seiten einige gute Ideen entdecken, die aber wegen der vielen anderen Effekte völlig untergehen. Es ist wie auf einem Marktplatz, auf dem alle Händler schreien, aber man kauft dann doch nichts. Renner stellt schon Anfang des 20. Jahrhunderts treffend fest: **Jedes Zuviel richtet den gleichen Schaden an wie ein Zuwenig.** Wenn Sie eine gute Idee haben, dann konzentrieren Sie sich auf diese. Nichts ist schlimmer, als den Besucher mit unnötigem visuellem Geflacker zu belästigen. Stil ist richtiges Weglassen des Unwesentlichen!

Dazu gehört auch – Sie merken, ich sammle Zitate zur Typografie –, was Kurt Weidemann sagt: **Gute Typografie erklärt den Inhalt, nicht den Gestalter.** Jeder Effekt, den Sie auf Ihrer Seite verwenden, muss seine Berechtigung haben. Nicht nur, weil es toll ausschaut oder weil man das eben mit Flash machen kann; so was macht man vielleicht auf einer privaten Seite oder einer Seite, die sich mit Flash-Kunst auseinandersetzt. Für alle anderen Einsatzgebiete gilt: Effekte haben den Sinn, auf etwas hinzuweisen, den Inhalt zu interpretieren oder selber eine »Geschichte« zu erzählen. Der Inhalt sollte durch die Äußerlichkeiten hindurchscheinen.

Auf der Seite 290 im Teil »How to« haben wir für Sie einen »Typografischen Sitecheck« durchgeführt.

Zusammenfassung

- Vorsicht! Wer eine (typografische) Regel bricht, macht zunächst nichts Besonderes. Er bricht nur eine Regel.
- Ist die Gestaltung meiner Flash-Seite dem Produkt/dem Kunden/den Benutzern angemessen?
- Kann der Text problemlos aufgenommen werden?
- Regt die Gestaltung zum Lesen an?

Animation mit Typografie

Was ist Flash-typischer als Texteffekte? Gerade in den letzten Monaten sieht man sie immer häufiger: Schriften, die Buchstabe für Buchstabe ins Bild laufen und dabei die schönsten Effekte erzeugen. Solche Effekte lassen sich auf verschiedene Art und Weisen erstellen. Handgetweente Typo-Effekte aber sind sogar Flash 1-kompatibel (wenn man auf Farbeffekte verzichtet).

Texteffekte tweenen

▶ Carlo Blatz

Für manche Effekte ist Handarbeit doch das Beste: für wenige Buchstaben, für Abwärtskompatibilität und für framesynchrone Animationen.

Handarbeit hört sich aufwändig an – ist es auch – aber es gibt Tricks, das zu beschleunigen. Das System ist denkbar einfach! Aus jedem Buchstaben wird ein Symbol gemacht und auf jeweils eine eigene Ebene gelegt.

Jeden Buchstaben in ein Symbol zu konvertieren, ist schon zeitintensiv. Noch aufwändiger ist es, diese Symbole wieder zu einem Wort zusammenzubauen. Etwas beschleunigen kann man das Erstellen der Symbole, indem man das geschriebene Wort in Vektoren zerlegt. So kann man jeden Buchstaben einzeln anklicken und in ein Symbol zu konvertieren, ohne die Positionen zu verändern. Mit den Shortcuts kann man die Buchstaben jetzt recht schnell auf einzelne Ebenen verteilen, einfach alle Buchstaben markieren und folgende Tastenkombinationen klicken: STRG+X (alle ausschneiden), auf die erste Ebene klicken, STRG+⇧+V (platziert einfügen), ⇧-Klick auf den ersten Buchstaben (einzelnes Objekt demarkieren), STRG+X, auf die nächste Ebene, STRG+⇧+V etc.

Wenn alle Buchstaben als Symbole auf jeweils einer eigenen Ebene liegen, erstellt man für alle »Zwischenstufen« der Animation über alle Ebenen neue Schlüsselbilder. Wenn alle Zwischenstufen verändert sind, z. B. skaliert, werden alle gemeinsam animiert – über alle Ebenen markieren und Bewegungstween aktivieren. Damit die Animation nun Buchstabe

für Buchstabe abläuft, müssen entsprechend die einzelnen Animationen in der Zeitleiste versetzt von einander starten und enden.

Tipp: Mit gut gesetzten Verzögerungen werden diese Animationen deutlich besser.

Texteffekte als Skript

Der Vorteil liegt auf der Hand. Man erstellt ein Skript, mit dem man fortan ganze Texte automatisch Buchstabe für Buchstabe animieren kann. Nun verwendet man vielleicht nicht ein und denselben Effekt für mehrere Kunden, aber wenn man einmal das System erstellt hat, genügen marginale Änderungen, um den ganzen Effekt zu verändern. Einziger Nachteil (wenn man das so nennen will) ist, dass der Effekt nur ab Version Flash 5 sichtbar ist, wohingegen per Hand getweente Effekte (ohne Alphaeffekt) sogar mit Flash 1 (Futuresplash) kompatibel sind.

◀
Saban Ünlü, www.netTrek.de, Carlo Blatz

Die Theorie

Durch eine Texteingabe definiert man in einer Variablen den zu animierenden Text. Da dieser Text Buchstabe für Buchstabe animiert werden soll, müssen wir die Länge des Textes auslesen und jeden Buchstaben einzeln in ein Textfeld schreiben, das wir dann animieren. Dieses Textfeld muss sich in einem MovieClip befinden, den wir duplizieren können. Die Animation des Textfeldes befindet sich in eben diesem MovieClip. Jede duplizierte Instanz muss nun aber um einen Buchstaben versetzt werden, da wir sonst kein Wort, sondern einen übereinander gelagerten Haufen Buchstaben erhalten würden.

Wir ermitteln also nach jeder Animation die Endposition X des Buchstabens und versetzen den nächsten MovieClip um einen Wert (Breite). Damit wir auch Zeilenumbrüche haben können, fragen wir das Zeichen »*« ab. Wenn im Wort dieses Zeichen gefunden wird, wird ein Zeilenumbruch erzwungen. Spätestens wenn die x-Position über den Wert 540 steigt, erzwingen wir den Zeilenumbruch ebenfalls. Der nächste Buchstabe muss nun an die x-Position des allerersten MovieClips (»Vorlage«) gesetzt werden, aber um einen Wert tiefer als der letzte Buchstabe – eine Zeile tiefer.

Die Praxis

Als Erstes erstellt man ein dynamisches Textfeld »buchstabe«, das in eine Filmsequenz (hier »buchstabe« genannt) eingebettet wird. Wichtig ist, dass alle Font-Outlines in das Textfeld eingebettet werden, damit wir später beliebige Animationen damit erstellen können. Der MovieClip bekommt nun den Instanznamen `mc_Buchstabe`. Diese Instanz wird in einem weiteren MovieClip »Vorlage« wie gewünscht animiert. Am Ende der Fade-in-Animation wird ein STOP gesetzt. Einige Frames später wird die Animation umgekehrt – Fade-out.

Diesen MovieClip legen wir nun auf die Bühne auf die Ebene »Vorlage« in Frame 2 und geben ihm den Instanznamen `mc_Buchstabe`.

In Frame 1 von Ebene »Button&Textfeld« liegt ein Textfeld, in dem der zu animierende Text eingegeben werden kann. Der Variablenname, der dem Textfeld zugeordnet ist, heißt: `text`. Weiterhin befindet sich hier ein Button, der auf Mausklick oder ↵-Taste den Film fortfahren lässt.

In Frame 1 von Ebene »Action« definieren wir nun die im Verlauf wichtigen Arrays, Variablen und Funktionen, auf die das Scripting zugreifen können muss.

```
stop ();
```

Nachdem der Film gestoppt wurde, wird ein Fokus auf das Textfeld gesetzt, damit man nicht erst dort hineinklicken muss, um darin schreiben zu können.

```
Selection.setFocus("_root.text");
```

Nun werden vier Arrays definiert, mit deren Hilfe man später dynamisch den Abstand zwischen zwei Buchstaben ermitteln kann. Der Grund dafür liegt auf der Hand, betrachtet man den Buchstaben i und den Buchstaben W, so ist das fast drei Mal breiter als i. Daraus folgt, dass der Abstand zwischen den Buchstaben WW und ii nicht identisch sein kann.

Also legt man die Buchstaben, die mit ihren Breiten in etwa übereinstimmen, in einem Array zusammen. Je mehr Arrays und folglich »Breiten«-Gruppen man anlegt und je mehr Zeichen in diese Gruppen eingeordnet werden, desto genauer ist dann später die Ermittlung der Buchstabenabstände.

```
liste1 = new Array();
liste2 = new Array();
liste3 = new Array();
liste4 = new Array();
liste1 = ["1","2","3","4","5","6","7","8","9","0","J","?","Z",
"S","F","L","q","e","z","u","o","p","ü","a","s","d","g","h","k",
"ö","ä","y","x","c","v","b","n","§","$","=","#","_","^","+","ß",
"_","<",">","µ"];
liste2 = ["E","R","T","U","P","Ü","A","D","Ö","Ä","Y","X","C",
"V","B","N","Q","G","H","K","w","%","&","O"];
liste3 = [".",":",";","!","'","´","`","i","l","j",",","  ","t",
"I","²","³","{","[","]","}"];
liste4 = ["r","f","/","(",")","-","°","|"];
```

Nun definieren wir die Funktion `ermittleBreite`. Diese wird später mit der Übergabe eines Parameters `Buchstabe` dazu benutzt, die Breite des übergebenden Zeichens zu ermitteln.

Dafür verschachtelt man zwei FOR-Schleifen. Die äußere sorgt dafür, dass alle Arrays durchgearbeitet werden, und die innere, dass alle Elemente des Arrays bearbeitet werden. In der inneren Schleife prüfe ich dann, zu welcher Breiten-Gruppe der Buchstabe gehört, und setze den Variablenwert von `breite` entsprechend.

Sollte das Zeichen nicht Element einer der Gruppen sein, so wird die zu Beginn vordefinierte Breite = 32 nicht verändert. D. h. durch `Return Breite` wird der Standardwert für nicht definierbare Breiten als Ausgabe der Funktion definiert.

```
function ermittleBreite (Buchstabe) {
   breite = 32;
   for (index1=1; index1<5; index1++) {
      liste = this["liste"+index1];
      for (index2=0; index2<liste.length; index2++) {
         if (index1 == 1 && liste[index2] == Buchstabe) {
            breite = 20;
         } else if (index1 == 2 && liste[index2] == Buchstabe) {
            breite = 26;
         } else if (index1 == 3 && liste[index2] == Buchstabe) {
            breite = 11;
         } else if (index1 == 4 && liste[index2] == Buchstabe) {
            breite = 13;
```

```
        }
      }
   }
   return breite;
}
```

Durch die Funktion **berechneBreite** wird beim Aufruf dynamisch der Abstand zwischen den übergebenden Breiten ermittelt. Dabei ist **breite1** der Parameter der Funktion, der die Breite des zuvor animierten Zeichens beinhaltet, und **breite2** die Breite des zu animierenden Zeichens. Um die Abstände jetzt schriftgerecht zu berechnen, muss man die Hälfte der **breite1** mit der von **breite2** addieren.

```
function berechneBreite (breite1,breite2) {
   return (breite1/2)+(breite2/2);
}
```

In Frame 2 von Ebene »Action«:

Machen wir unsere Vorlage unsichtbar und definieren die Variable **position** vor, mit der wir später das Zeichen definieren, das animiert werden soll. Weiterhin ermittelt man die Zeichenanzahl des zu animierenden Texts und setzen dies als Variablenwert von **anzahlBuchstaben** ein.

```
_root.mcVorlage._visible= 0;
position = 0;
anzahlBuchstaben = length(_root.text);
```

In Frame 3 von Ebene »Action« wird die Positionierung des zu animierenden Buchstabens durchgeführt. Wir erweitern die Variabel **position** um den Wert 1. Dadurch animieren wir jeweils das nächste Zeichen. Damit wir dieses Zeichen auch unterbringen können, wird die Vorlage dupliziert. Deren Instanzname wird auf **buchstabe + position** gesetzt und die Tiefe auf Position. Beim ersten Durchlauf, d.h. bei der Animation des ersten Zeichens würde dies bedeuten, dass Instanzname = **buchstabe1** und Tiefe = 1 ist.

Das zu animierende Zeichen setzen wir in die Variable **aktuellerBuchstabe**, indem wir exakt die derzeitige Position aus der Variable **text** entfernen. Dies macht man mit **charAt()**, der Parameter, der dieser Funktion übergeben wird, gibt den Index an, der das auszuschneidende Zeichen definiert.

Da das erste Zeichen Index = 0 ist, aber die Variable `position` in diesem Frame mit dem Wert 1 beginnt, muss der Parameter als `position -1` definiert werden.

```
++position;
_root.mcVorlage.duplicateMovieClip(["buchstabe"+position], posi-
tion);
aktuellerBuchstabe = _root.text.charAt(position-1);
```

Wenn das Zeichen, das animiert werden soll, ein Stern ist, soll der duplizierte MovieClip unsichtbar gemacht werden und die Variable `zeilenumbruch` als `true` definiert sein.

```
if (aktuellerBuchstabe == "*") {
   this["buchstabe"+position]._visible = false;
   zeilenumbruch = true;
```

Sonst wird die im duplizierten MovieClip befindliche Variable `buchstabe` auf den Variablenwert von `aktuellerBuchstabe` gesetzt.

Weiterhin wird geprüft, ob es sich um das erste zu animierende Zeichen handelt, und falls ja, wird dessen duplizierter Clip auf dieselbe x-Position der Vorlage gesetzt.

```
} else {
   this["buchstabe"+position].mc_Buchstabe.buchstabe = aktueller
Buchstabe;
   if (position == 1) {
      this["buchstabe"+position]._x = _root.mcVorlage._x;
```

Falls es nicht das erste zu animierende Zeichen ist, wird geprüft, ob die Variable `zeilenumbruch` als wahr definiert wurde oder die x-Position des zuvor duplizierten MovieClips nicht 540 ist bzw. diesen Wert überschreitet.

Wenn das zutrifft, muss jetzt das zu animierende Zeichen eine Zeile tiefer gesetzt werden und folglich die y-Position des neuen MovieClips 35 Pixel tiefer liegen als die y-Position der zuvor erstellten Zeile.

Die x-Position muss gleich der x-Position der Vorlage sein, damit die Zeilen untereinander liegen. Zur Vorbereitung eines eventuell folgenden Zeilenumbruchs wird die Variable `zeilenumbruch` zurückgesetzt auf `false`.

```
   } else {
      if (this["buchstabe"+(position-1)]._x>=540 or zeilenumbruch) {
         this["buchstabe"+position]._x = _root.mcVorlage._x;
         this["buchstabe"+position]._y = this["buchstabe"+(position-
2)]._y+35;
         zeilenumbruch = false;
      } else {
```

Ansonsten ist die x-Position des zuvor zu animierenden Clips kleiner 540 und `zeilenumbruch` = `false`.

Wird die x-Position des neuen MovieClips definiert durch die Ausgabe der Funktion `ermittleBuchstabe`, so wird er addiert mit der x-Position des zuvor animierten MovieClips.

Die y-Position entspricht der des zuvor animierten MovieClips.

```
         vorherigerBuchstabe = _root.text.charAt(position-2);
         this["buchstabe"+position]._x = this["buchstabe"+
(position-1)]._x+_root.berechneBreite(_root.ermittleBreite
(aktuellerBuchstabe), _root.ermittleBreite(vorheriger
Buchstabe));
         this["buchstabe"+position]._y = this["buchstabe"+
(position-1)]._y;     }
   }
}
```

Frame 4 vom Ebene »Action«: Es wird geprüft, ob noch nicht alle Zeichen animiert wurden (`position < anzahlBuchstaben`). Wenn dies der Fall ist, geht man zurück in Frame 3, um das nächste Zeichen zu animieren. Ansonsten setzt man die Variable `position` zurück auf null und lässt den Film weiterlaufen.

```
if (position < anzahlBuchstaben) {
   gotoAndPlay (_currentframe-1);
} else {
   position = 0;
}
```

Frame 12 von Ebene »Action«: Die `position` wird um 1 erhöht.

```
++position;
```

Frame 13 von Ebene »Action«: Wenn noch nicht alle Zeichen ausgeblendet wurden (dies ist der Fall, wenn `position <= anzahlBuchstaben`), dann wird das derzeitig aktuelle Zeichen `position` oder besser gesagt dessen MovieClip dazu veranlasst, zur Bildbezeichnung weiter zu gehen und von hier den Clip abspielen zu lassen. Anschließend gehen wir zurück zu Frame 12 um `position` aufzählen zu lassen. Wenn aber alle Zeichen ausgeblendet sind, soll der MovieClip zum nächsten Bild gehen und den Film stoppen.

```
if (position <= anzahlBuchstaben ) {
   _root["buchstabe" + position].gotoAndPlay("weiter");
   gotoAndPlay (_currentframe-1);
} else {
   nextFrame ();
}
```

Der Schreibmaschineneffekt

Es ist möglich, den Schreibmaschineneffekt auf drei verschiedene Arten zu simulieren.

◀
Carlo Blatz

Erstellung

1. Man erstellt eine Einzelbildanimation, die jeweils um einen Buchstaben ergänzt wird. Das ist Flash 1-kompatibel, aber die Dateigröße leidet darunter.
2. Man arbeitet mit einer Maske, die einen Balken, der den Text verdeckt, langsam verschwinden lässt.
3. Mit ActionScript – also ab Flash 4: Dies ist die beste und auch einfachste Lösung, da sie – einmal eingerichtet – schnell zu verwenden und auch schnell wiederzuverwerten ist. Im Gegensatz zu Nr. 1 ist der Text auch im Nachhinein editierbar.

Erstellung per ActionScript

Man erstellt zwei Ebenen. In die Erste setzen wir das Textfeld (fünf Frames lang), in die zweite die Steuerung mit vier Keyframes.

Keyframe 1 in Frame 1:

Der Inhalt des Textfeldes »TypeField« wird in die Variable `Textbuffer` geschrieben und die Variable `TextLen` auf 1 gesetzt. Im Textbuffer speichern wir den Inhalt des Textfeldes zwischen, da wir das Textfeld gleich auch für den Typeeffekt verwenden. `TextLen` ist unser Buchstabenzähler.

```
TextBuffer = TypeField;
TextLen = 1;
```

Keyframe 2 in Frame 6:

Dieser Keyframe sitzt sechs Frames versetzt, damit wir einen Zeitbuffer haben, mit dem wir erreichen, dass die neuen Buchstaben nicht linear animiert werden, sondern – wie das menschliche Tippverhalten – zeitlich nichtlinear.

Wenn `TextLen` kleiner oder gleich der Länge unseres Textbuffers ist und ungleich 0, wird der nächste Buchstabe animiert. `TextLen` wird mit 1 addiert, und ins Textfeld wird der nächste Buchstabe hinzugefügt. Danach springt Flash zufällig in ein Frame zwischen dem zweiten und dem sechsten Frame, wodurch die Zeit bis zu Erscheinen des nächsten Buchstabens unregelmäßig ist.

```
If (TextLen<=TextBuffer.length && TextLen!=0){
   TypeField = TextBuffer.substring (0, TextLen);
       TextLen++;
   gotoAndPlay (Random (2)+4);
}Else{
   TextLen = 0;
}
```

Andernfalls wird `TextLen` auf 0 gesetzt, und die Bedingung ist im nächsten Durchlauf nicht mehr wahr.

Der Schreibmaschineneffekt funktioniert nun bereits, allerdings sieht zu Anfang man für den Bruchteil einer Sekunde den ganzen Text, bis er anfängt, sich selbst aufzubauen. Um das zu verhindern, legt man den ganzen Effekt in eine Filmsequenz und macht diese beim Start unsichtbar (`MC._visible = false`). Das Skript macht ihn dann erst wieder sichtbar, wenn `TypeField` nicht mehr dem `TextBuffer` entspricht. Eine andere Möglichkeit wäre es, die Variable aus dem Skript zu holen und nicht aus dem Textfeld.

◀ **Abbildung 1**
Tweening der Maske über dem Textfeld

Maskentexteffekt mit Duplicate Movie

◀
Carlo Blatz, Hassan Beigi, www.powerflasher.de

Hier wird eine Filmsequenz dupliziert. Auf der Hauptzeitleiste liegt nur ein Frame mit der Aktion

```
Anim1.Text = "Powerflasher";
```

Eine Ebene darunter liegt eine Filmsequenz **Anim**, in ihr befinden sich vier Frames mit Skripten und eine Filmsequenz **FX**. In dieser wiederum liegt eine Filmsequenz **Mask** und darin der letzte MovieClip **Text**.

Fangen wir also von unten an. In **Text** befindet sich ein Textfeld **Text** und zwei Frames. Im letzten Frame ist die Aktion

```
Text = _parent._parent._parent.Text;
this.stop();
```

Er holt sich also aus der Ebene des MovieClip **Anim** (drei MovieClips tiefer) die Variable für das Textfeld. Diese wird später von der Hauptzeitleiste aus an **Anim** gesendet.

Dieser MovieClip **Text** ist nun von **Mask** maskiert. Ein schmaler Streifen läuft als Maske von oben nach unten über den Textfeld-MovieClip. Diese Animation geschieht allerdings erst im zweiten Keyframe. Das erste ist leer und gestoppt – damit es unsichtbar ist, bis er die Aktion **play** bekommt.

Dieser MovieClip **Mask** liegt nun wie gesagt im MovieClip **FX** und hier befindet sich die eigentliche Animation. Der MovieClip **Mask** wird mit ei-

ner Verzögerung herunter und dann wieder heraufgetweened. Am Ende wird die Variable `K` hochgezählt:

```
_parent.k++;
this.stop();
```

Dieser MovieClip kann nun dupliziert werden. Damit die Animation später auf beliebigen Zeitleisten eingesetzt werden kann, liegt das eigentliche Skript im MovieClip `Anim`. Hier wird im ersten Keyframe der eigentliche MovieClip `FX` unsichtbar gemacht, denn er soll ja gleich dupliziert werden. Die Variablen i und k werden mit dem Wert 1 initialisiert.

```
this.i = 1;
this.k = 1;
this.FX._visible = false;
```

Im zweiten Keyframe (Bezeichnung »Duplicate«) liegt nun die Duplicate-Aktion.

```
this.FX.duplicateMovieClip (["FX"+this.i], 900+this.i)
```

Ein Keyframe weiter (3) wird die Maske in der neu duplizierten Filmsequenz angewiesen, im Vergleich zur vorherigen Filmsequenz einen Frame weiter zu springen. Das geschieht durch die Laufvariable `i`. Sie wird immer um 1 größer, somit wirkt sich das auf die neue Filmsequenz (`FX +this. i`) und auf das nächste Keyframe aus (`gotoAndStop(i+1);`). Nun wird `i` um 1 erhöht und zurück zum Label »Duplicate« (Keyframe 2) gesprungen, solange `i` kleiner ist als 23.

```
this["FX"+this.i].Mask.gotoAndStop(i+1);
this.i++;
this.i<23 ? gotoAndPlay ("Duplicate") : gotoAndStop ("Ende");
```

Der MovieClip wird also 23 Mal dupliziert. Ist das geschehen, springt er zu `"Ende"`:

```
this.stop();
```

▲ Abbildung 2
Maskentexteffekt

Von der Hauptzeitleiste schreibt man den Inhalt des Textfeldes mit der Aktion:

```
Anim1.Text = "Powerflasher";
```

`Anim1` ist der Instanzname vom MovieClip `Anim` auf der Hauptzeitleiste.

Diese Animation kann man jetzt schnell variieren, indem man einfach die Animation im MovieClip `FX` verändert. So könnte man z. B. noch Farbeffekte einbauen.

Sound

Der gute Ton gehört dazu …

Tja, und wie … Jeder Flasher kennt den Moment, wenn die Animation zum aller ersten Mal mit Musik abläuft. Plötzlich wirkt alles anders, plötzlich lebt das Ganze. Trotzdem wird mit Sounds oft bloß oberflächlich umgegangen, und trotzdem trifft man immer wieder Seiten, wo nur »irgendein Loop« genommen wurde. Das Klangerlebnis kann einem Flash-Film Flügel verleihen oder eben diese stutzen.

▸
Geza Varkuti

Verfügbarkeit

Online-Quellen

Ohne Zweifel ist es das Einfachste, wenn der Flasher zur Tonkonserve greift. Fertiges Material findet man im Web in Hülle und Fülle.

Sound im Netz
Hier einige Sites, die Sounds kostenlos zur Verfügung stellen oder Sound-Bundles verkaufen: www.flashworker.de; www.partylogger.de; www.soundworker.de; www.flash4all.de; www.flashkit.com; www.flashsounds.de; www.music4flash.com

Jede Site bietet natürlich auch hochwertige Sounds an. Also sind vor allem Geduld und Ausdauer angesagt, weil man sich bei der Suche nach den Wunschtönen schließlich etliche Sounds anhören muss. Die Strukturierung erfolgt in der Regel nach Verwendungsgebieten wie Buttons, Loops, Effekte etc. Von den Non-Profit-Sites wie www.flashworker.de, aber auch von den kommerziellen Sites wie www.esoundcity.com bieten einige auch eine Stichwortsuche an (eine kleine Kollektion von eSoundCity finden Sie auf der beiliegenden CD unter Gestaltung/Sound/Waves). Bei der Suche kann man hier enorm Zeit sparen.

Offline-Quellen

Die zweite Quelle für Sounds sind die Sammler-CDs. Die bekanntesten in Deutschland sind Monster Pack (DM 49) von Data Becker und Soundcube (DM 99) von Bestservice. Beide sind sehr empfehlenswert.

Monster Pack liefert acht CDs, die Sounds sind im Dateiformat .wav vorhanden. Die CDs sind in folgende Kategorien eingeteilt: Techno, House, Hiphop, Dance, Rock/Pop, Klassik/New Age, Vocals und Effekte. Ein ganz besonderes Highlight der Sammlung sind sicherlich die Effekte. Sie sind unterteilt in Bereiche wie z. B. Machines, 3D fx, Percs. Wenn man Sounds für ein Intro, den Preloader oder für einen einfachen Button sucht, wir man hier mit Sicherheit fündig.

Soundcube bietet zehn CDs, allerdings sind die Dateien in drei Dateiformaten vertreten (schließlich gibt es auch das Volk der Mac-User). Die CDs sind in folgende Kategorien unterteilt: Dance Instruments, Orchestra/Classic, Voice Spectral, World FX, Crash Boom Zap, Soundtracks, Ethnic Journey und Race X. Auf den CDs findet man wirklich alles, was das Flash-Herz begehrt, egal, ob man OneShots, Loops oder Sounds für ein Intro sucht.

Zehn CDs für knapp DM 100 – hört sich gut an. Aber wenn ich bedenke, dass z. B. ein PC-Benutzer nur die WAV-Files braucht, sind es plötzlich nur noch ca. vier CDs, und wenn man die unbrauchbaren Sachen (z. B. die Zahlen auf Englisch usw.) abzieht ... Na ja, dann sieht das Preis-Leistungs-Verhältnis schon etwas anders aus.

Ein bisschen Theorie

Wenn wir schon über verschiedene Dateiformate sprechen, lassen Sie uns dieses Thema etwas detaillierter behandeln. Es ist wichtig, dass Sie wenigstens ein bisschen von dessen Theorie verstehen und die darin liegenden Möglichkeiten und Grenzen erkennen.

Bis Flash 4 waren die einzig möglichen Dateien, die importiert werden konnten, die mit der Endung .wav für den PC und .aiff für Mac.

Flash 5 ist nun in der Lage, auch MP3-Dateien zu integrieren.

Drei Attribute bestimmen hauptsächlich die Qualität (und gleichzeitig die Größe) einer Sounddatei:

1. Stereo/Mono
2. Die Sampling-Rate
3. Die Sampling-Auflösung

Stereo versus mono

Stereo oder Mono ist das am einfachsten zu erkennende Attribut. Man verzichtet leider oft auf Stereo, weil dies zu einer doppelten Dateigröße führt. Viele verstehen unter stereo nur die üblichen Panoramaspielereien: Gitarre – rechts, Klavier – links. Man kann aber auch ein einziges Instrument stereo aufnehmen, dabei wird der Raum, in dem es spielt, »mit aufgenommen« oder später mit einem Stereohall versehen. Dies klingt viel besser als eine Punktquelle.

Leider ist aber die Zeit für solche ästhetischen Überlegungen in Verbindung mit Flash noch nicht reif. Wir bräuchten viel schnellere Leitungen, um diese Sounds benutzen zu können. Wenn es dramaturgisch sinnvoll ist, kann man vielleicht einen sehr kurzen Stereo-Raum-Effekt einbauen, aber möglichst ohne die Leitung zu sprengen.

Sampling-Rate

Das zweite Attribut ist die so genannte Sampling-Rate. Ein Ton wird digitalisiert, indem man in bestimmten periodischen Zeitabständen ein Muster und in digitale Daten konvertiert. Je mehr Muster pro Sekunde zur Verfügung stehen, desto genauer wird der Ton nachgebildet, aber selbstverständlich verbraucht die erzeugte Datei auch umso mehr Speicherplatz. Die Sampling-Rate entspricht dieser »Häufigkeit« in Hertz. Die CD-Qualität ist 44,1 kHz (also werden 44.100 Muster innerhalb einer Sekunde genommen), aber es gibt schon heute Sampling-Raten, die deutlich darüber liegen. Manche Soundkarte und Software ist schon in der Lage, mit 96 kHz zu sampeln.

Dateien mit 22 kHz besitzen die so genannte Radioqualität. Sie sind sehr gut zu gebrauchen. Darunter ($\leq$ 11 kHz) sind die Qualitätseinbußen nicht zu überhören – für Musik sind sie nicht mehr empfehlenswert, obwohl man manche Effekte auch noch mit dieser Sampling-Rate verwenden könnte.

Ein Klang ist nichts anderes als die Veränderung des Luftdrucks. Die Schwingungen werden in Hertz gemessen. Der mittlere A-Ton entspricht 440 Hertz, mit jedem Oktavsprung verdoppelt sich diese Zahl, so dass der höchste Ton auf dem Klavier ca. 4 kHz entspricht. Unser Gehirn kann Klanginformationen ab 20 Hz bis ca. 20 kHz auswerten, also Schwingungen, die sich zwischen 20 und 20.000 Mal pro Sekunde wiederholen. Es besteht ein wichtiger Zusammenhang zwischen Sampling-Rate und Tonhöhe.

Faustregel: Man braucht mindestens die doppelte Sampling-Rate, um einen Ton richtig nachzubilden. Das heißt, wenn wir bis 20 kHz hören können, brauchen wir für eine gute Aufnahme mindestens 40 kHz Sampling-Rate (daher CD-Qualität = 44,1 kHz).

Ein Klang in der Natur ist immer eine ziemlich chaotische Ansammlung verschiedenster Frequenzen. Wie diese Frequenzwerte aufeinander folgen, wie ihre relative Lautstärke ist – all das kann unser Gehirn wahrnehmen, auswerten, und wir wissen dann, ob jemand in einer Höhle oder in einem Konzertsaal geigt.

Ein Telefon ist zu einem Frequenzgehalt von 400 bis 8000 Hz fähig. Schon die einfachsten Soundkarten sind in der Lage, 20 bis 20.000 Hz zu verarbeiten.

Sampling-Auflösung

Wir haben gesagt, dass beim Sampeln aus dem Klang in bestimmten Abständen Muster genommen werden. Wie viele Informationen in diesem einzelnen Sample gespeichert werden können, bestimmt die Sampling-Genauigkeit oder Sampling-Auflösung. Stellen Sie sich dies bitte ungefähr wie die Farbenauflösung vor. Die am häufigsten benutzten Auflösungen sind 8 Bit, 16 Bit und 24 Bit, was den Zahlen 256, 65.535 (HighColor) und 16 Millionen (TrueColor) entspricht. Im Gegenteil zu den Farben, die bei einer Auflösung von 256 Farben ein recht gutes Bild liefern können, klingen 8-Bit-Sounddateien eher schlecht.

MP3

Was bei den Grafikdateien *.jpg, ist bei Sounddateien MP3. Kurz gesagt: ein hochkomprimiertes Dateiformat für Audiodateien. Wenn man die CD-Qualität beibehalten will, erreicht man eine Komprimierungsrate von ca. 12:1. Also braucht man für eine Minute Musik nur ca. 1 MB (mit stärkerer Komprimierung natürlich noch weniger). Die Komprimierungsmethode stammt übrigens aus Deutschland, sie wurde von Dr. Karlheinz Brandenburg am Fraunhofer-Institut unter der Schirmherrschaft von Motion Picture Expert's Group (daher das Kürzel) ausgearbeitet.

Da wir nur ein Spektrum von 20 bis 20.000 Hz hören können, beginnt der MP3-Encoder die Komprimierung mit dem Löschen von Tönen, die wir sowieso nicht hören können. Danach untersucht er den Tonspekt-

rumablauf. Wenn er gleichzeitig auf der selben Frequenz mehrere Töne findet, speichert er nur den lautesten. Der überdeckt die leiseren Töne schließlich. Das ist natürlich eine sehr vereinfachte Darstellung des Vorgangs, aber im Großen und Ganzen wird so vorgegangen.

Abgesehen von den Audioinformationen können Sie in einem MP3-File zusätzliche Informationen zu dem Musikstück speichern, z. B. Titel, Künstler, Copyright etc.

Damit die Verwirrung vollkommen wird, benutzt MP3 für den Ausdruck der Sampling-Rate eine andere Maßeinheit: kbps (Kilobits pro Sekunde).

64 kbps ergeben eine passable Qualität, 80 kbps sind etwas besser, 128 kbps ist ungefähr die CD-Qualität (zumindest wird sie vom MP3-Programmhersteller so bezeichnet), und 160 kbps werden sogar als Oversampling bezeichnet. Man sollte auf jeden Fall bei jedem Musikstück etwas experimentieren, um die optimale Komprimierungsrate zu finden.

Flash ist in der Lage, WAV-Dateien für den Flash-Film selbst auf MP3 zu komprimieren. Sollten Sie einmal MP3 außerhalb von Flash konvertieren, besorgen Sie sich am besten einen so genannten Ripper. Der wohl bekannteste ist MusicMatch Jukebox. In der Standardversion können Sie ihn kostenlos von www.musicmatch.com herunterladen. Diese Version lässt aber nur eine Sampling-Rate von 96 kbps und weniger zu, die registrierte Vollversion kostet dann 30 $.

Sounds aufnehmen und bearbeiten

Wenn jemand mit den Konservensounds nicht ganz zufrieden ist, kann er sie bearbeiten oder mit dem nötigen Know-how selber erstellen. Bitte bedenken Sie auch, ob Sie für Ihren Kunden nicht besser einen individuellen Sound erstellen: Stellen Sie sich ein Mal vor, Sie machen einen Flash-Film mit Konservenmusik für Mercedes, und später findet der Auftraggeber denselben Sound bei dem Hundeverein Peterstal wieder (oder noch schlimmer: bei BMW).

Was braucht man, um Sounds mit dem Computer zu bearbeiten oder aufzunehmen?

Aufgepasst, jetzt wird es richtig teuer! Zuallererst benötigen wir die nötigen Hardwarekomponenten. Einen besonders flotten Rechner brauchen Sie nur dann, wenn Sie größere Musikdateien bearbeiten, sonst tut es auch der normale Arbeitsrechner. Die Festplatte sollte dementspre-

chend dimensioniert sein. Die folgende Formel hilft Ihnen, die nötige Kapazität zu ermitteln:

Spurenzahl x Auflösung in Bit x Sampling-Rate in Hz, dividiert durch 8, dividiert durch 1024 und noch einmal dividiert durch 1024 = MB/Sec.

So benötigt z.B. eine Aufnahme in Stereo mit 16 Bit Auflösung und 44.100 Hz Sampling-Rate:

2 x 16 x 44100 : 8 : 1024 : 1024 = 0,168 MB/Sec.

Für den Arbeitsspeicher nehmen Sie so viel RAM, wie Sie bezahlen können, aber mindestens 32 MB.

Die Soundkarte

Schon die einfachsten Karten versprechen eine CD-Qualität, was aber leider meistens nicht der Wahrheit entspricht. Sie sind tatsächlich in der Lage, mit 44.100 Hertz aufzunehmen, aber die so genannten Analog-Digital-Wandler sind meistens minderwertig. Auch bei den Anschlussbuchsen müssen Sie sehr vorsichtig sein, sie oxidieren mit der Zeit; daher können Kontaktprobleme auftreten.

Wer ein bisschen Wert auf die Qualität seiner Sounddateien legt, sollte auch das Geld für eine hochwertige Soundkarte nicht scheuen. Ich werde den Moment niemals vergessen, als ich nach der Installation meiner ISIS den Rechner neu starten musste und der Microsoft-Sound ertönte – es war fast schockierend, und ich wusste sofort, dass die Investition richtig gewesen war.

Als Erstes nehmen wir vielleicht die **Turtle Beach Montego II**. Sie kostet ca. DM 400 und bietet zwei Line- und zwei Mikrofoneingänge sowie je einen digitalen Ein- und Ausgang (optisch und koaxial). Auch die MIDI-Anschlüsse sind vorhanden. Die Anschlussbuchsen sind vergoldet. Die klanglichen Eigenschaften sind sehr schön ausgeglichen und machen einen ausgesprochen angenehmen Eindruck. Ausgeliefert wird die Karte mit den Programmen Audio View und MIDI Orchestrator sowie verschiedenen Diagnoseprogrammen.

Die nächste Stufe kommt aus den Labors des französischen Klangschmieds Guillemot und heißt **ISIS**. Die Karte kostet um die DM 600, bietet aber deutlich mehr als die Montego.

Der Hersteller nennt sie bescheiden »8 Input/4-Output Hardware Vollduplex Multikanal Direct-to-Disk Audio und MIDI Studio«. Die Ein- und Ausgänge sowie der A/D-Wandler sind in einer externen Hardwarebox untergebracht – sehr praktisch, denn endlich muss man nicht mehr hinter den Rechner kriechen, außerdem hätten so viele Buchsen hinten sowieso keinen Platz. Die ISIS hat also acht Line-Inputs sowie zwei S/P DIF (Sony/Philips Digital Interface) Inputs (optisch und koaxial) und genauso viele Ausgänge. Für die Regelung der Ein- und Ausgänge ist eine Softwarekonsole zuständig.

Außer den üblichen GM/GS MIDI-Klängen bietet die ISIS die Möglichkeit, mit eigenen Sampels Soundbänke zu konfigurieren. Dazu stehen 4 MB RAM (erweiterbar bis auf 32 MB) zur Verfügung. Der Geräuschabstand bei der Aufnahme liegt bei 94 dB, womit die ISIS in den Bereich guter Hi-Fi-Digitalrecorder kommt.

Guillemot liefert hochwertige Softwarekomponenten mit. Der Wave-Editor CoolEdit Pro SE und Logic Audio Pro ISIS von Emagic sind wirklich eine erstklassige Wahl.

In die dritte Kategorie gehört die Soundkarte **EWS88 MT von Terratec** made in Germany. Das Audiosystem bietet 24 Bit Wandlerbausteine mit einer Sampling-Rate von bis zu 96 kHz. Der Signalrauschabstand beträgt im analogen Ausgang 108 dB. Die Karte wird ebenfalls mit einem externen Rack geliefert, indem die acht analogen und zwei digitalen sowie die MIDI-Ein- und Ausgänge untergebracht sind. Als Software werden Emagics microLogic und Samplitude Basic von SEKD beigepackt. Qualität hat sicherlich ihren Preis (ca. DM 1000), aber bei der EWS88 MT müssen Sie keine Kompromisse schließen. So, jetzt wissen Sie, welche Soundkarte die richtige für Sie ist!

Lautsprecherboxen

Die meisten Boxen, die bei einem Computerkauf mitgegeben werden, können Sie getrost in den Sondermüll tragen. Wer damit Musik hört oder sogar editiert, begeht ein schweres Attentat auf seine Ohren.

Entweder kaufen Sie sich Aktivboxen (Lautsprecher mit eingebautem Verstärker) oder Passivboxen, in diesem Fall brauchen Sie zusätzlich noch einen Verstärker.

Hier eine Empfehlung abzugeben ist sehr schwer, da Lautsprecher auszuwählen eine ausgesprochen individuelle Angelegenheit ist. Ich persönlich schwöre auf Tannoy, aber kenne Kollegen, die mit den Yamaha Aktiv-

boxen (MSP 5) oder mit dem L90 von JBL sehr gute Erfahrungen gemacht haben. Vielleicht doch ein Geheimtipp: Die nuBox 360 kommt aus dem Schwabenland und bietet ein unschlagbares Preis-Leistungs-Verhältnis. Sie sollten sich die Lautsprecherboxen vor dem Kauf natürlich unbedingt anhören.

Kopfhörer

Kopfhörer anstatt einer Stereoanlage mit Boxen einzusetzen, halte ich nicht für eine gute Idee. Mit einem guten Kopfhörer kann man sicherlich mehr Details kontrollieren, aber bei längerem Arbeiten kann es recht unbequem werden. Ob Sie einen geschlossenen oder einen halb offenen nehmen, ist Geschmackssache, hier gilt auch: erst ausprobieren. Wovon ich abraten möchte, sind die kabellosen Kopfhörer. Selbst teure Modelle liefern einen schlechteren Klang als die Artgenossen mit Kabel.

TV-Karte

Ich möchte hier noch eine Hardwarekomponente erwähnen, die scheinbar nichts mit Audiobearbeitung zu tun hat. Oft werden Geräusche bei Flash-Filmen eingesetzt, und ebenso oft wird danach gesucht. Mein Tipp: der Fernseher. Wenn jemand den Sound des Tennisschlags sucht, liegt es auf der Hand nachzuschauen, ob bei Eurosport, DSF oder anderswo nicht gerade ein Match übertragen wird. Natürlich kann die Verkabelung zwischen TV und Computer recht umständlich werden, deshalb rate ich in diesem Fall dazu, sich eine TV-Karte zu besorgen. Schon ab DM 100 bekommen Sie TV-Karten, die sogar die zeitgesteuerte Aufnahme zulassen. Auf diese Weise können Sie bequem zu guten und copyright-freien Geräuschen gelangen. Aber bitte keine Musik sampeln: Die ist immer urheberrechtlich geschützt!

Softwarekomponenten

Erst sehen wir uns die Sound- oder Wave-Editoren an und danach die kompletten MIDI-Audio-Studiolösungen. Selbstverständlich kann ich hier nicht alle Funktionen und die Arbeitsweisen vorstellen, dazu sollte man die Handbücher lesen, was ich Ihnen an dieser Stelle ans Herz legen möchte. Audioprogramme intuitiv zu erlernen nach dem Motto »Learning by Doing« (oder anders gesagt: »Trial and Error«) gestaltet sich viel

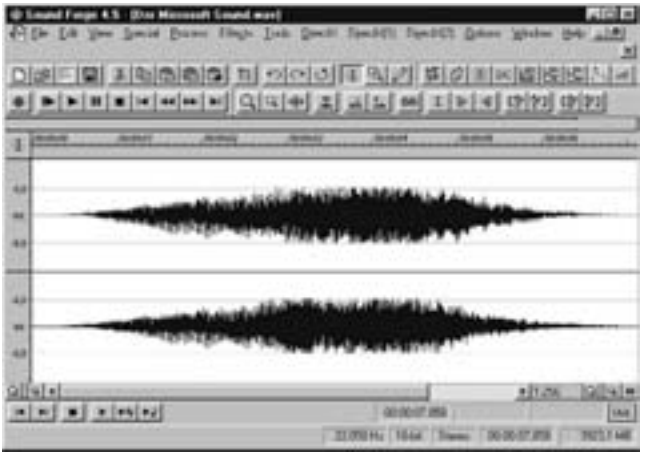
Abbildung 1 ▸
Die Goldwave-Oberfläche

schwerer als bei anderen Programmgattungen. Auf jeden Fall sollte man sich über eine Menge Fachbegriffe wie Trim, Normalize oder Noise Gate etc. im Klaren sein, um sie einsetzen zu können. Dazu kommt noch, dass manche Programme »Made in Germany« einer ziemlich eigenwilligen Logik folgen. Es geschieht nach einem Klick nicht immer gleich das, was man von einem Windows-Programm so erwarten würde. Mein Tipp: Handbücher lesen und immer weiter lernen!

Die untere Preiskategorie gehört Goldwave. Sie können sie bei www.goldwave.de für DM 90 erwerben.

Der Preis-Leistungs-Verhältnis ist hier mehr als o.k., das Programm ist leistungsfähig und bietet eine Menge Bearbeitungsmöglichkeiten und Effekte. Für Anfänger ist dies gewiss die beste Wahl. Goldwave unterstützt drei Arten von Datei-Handling.

- Bei Direct-to-Disk wird vom Sound eine temporäre Kopie angelegt, alle Änderungen werden in dieser Datei durchgeführt. Vorteil: kleinere RAM reichen aus; Nachteil: etwas langsamer, da alle Schritte auf die Festplatte geschrieben werden.
- Beim Modus-RAM werden die Änderungen im RAM durchgeführt. Vorteil: schnell; Nachteil: Die RAM-Kapazität bestimmt die Größe der Datei.



◄ Abbildung 2
Soundforge

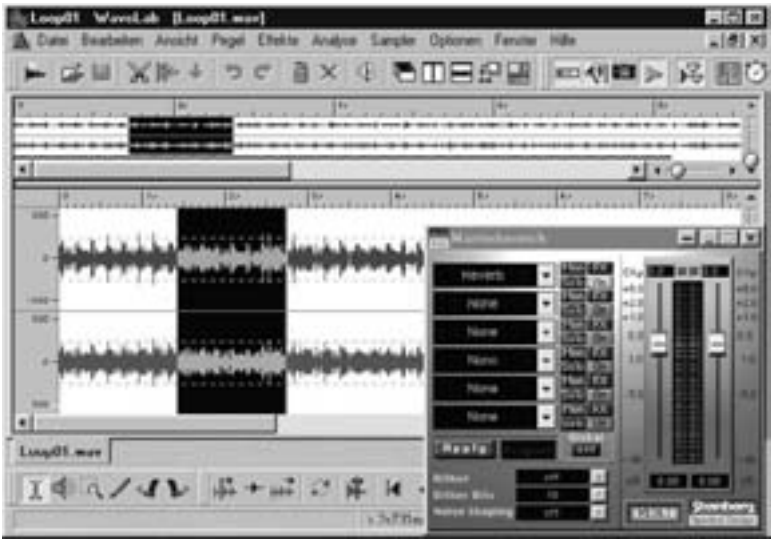
- Flash heißt die dritte Möglichkeit (hat nichts mit Macromedias Flash zu tun). Es verwendet einen gemischten Modus: Die ersten fünf Sekunden kommen über RAM, der Rest von der Festplatte. Der Haken: Sie benötigen einen schnellen Computer.

Für das Hinzufügen eines Effekts braucht Goldwave je nach Größe der Datei etwas Zeit. Wenn Sie z. B. ein Echo anwenden, wird der Sound neu berechnet, und wenn er fertig ist, können Sie sich das Ergebnis anhören. Sollte es Ihnen nicht gefallen, müssen Sie mit der Undo-Funktion die Änderungen rückgängig machen. Bei der täglichen Arbeit könnte diese Arbeitsweise eine Barriere darstellen. Batch-Bearbeitung ist leider nicht möglich, dafür gibt es aber eine Audio-Extrakt-Funktion. Sie können von Audio-CDs und von AVI-Filmen problemlos WAVs erzeugen.

Der erste Schritt in Richtung professioneller Audiobearbeitung ist das Programm Sound Forge vom amerikanischen Hersteller Sonic Foundry (DM 695).

Das Programm unterstützt Sampling-Raten bis zu 96.000 Hz und lässt Batch-Bearbeitung zu. Außer den integrierten Effekten unterstützt es die MS Direct-X Plug-in-Technologie, so kann man Sound Forge um Effekte von Drittherstellern erweitern. Bei der Stapelbearbeitung können Sie alle Funktionen und Direct-X Plug-ins anwenden. Mit der bei Windows üblichen Iconleiste ist dies eines der wenigen Soundprogramme, das sich intuitiv bedienen lässt (obwohl es nur in Englisch erhältlich ist). Sie können in nahezu jedem Dateiformat schreiben, einschließlich WAV, ASF, RM, MP3, WMA und AVI.

Abbildung 3 ▸
Wavelab



Hier noch ein paar interessante Features: Suchfunktion für Audioereignisse, mehrere Ebenen für das Rückgängigmachen und Wiederherstellen, professionelle Analysewerkzeuge und direkte Kommunikation mit Hardware-Samplern.

Das Topprogramm schlechthin ist WaveLab von der Firma Steinberg, auch in Deutsch erhältlich für DM 899.

Außer den Bearbeitungs- und Effektmöglichkeiten wartet WaveLab mit CD-Mastering (Brennen), Samplerunterstützung sowie mit einer Sounddatenbank (AudioAccess) auf. Das Programm kommuniziert mit allen handelsüblichen Samplern direkt via MIDI oder SCSI. Mit der Sounddatenbank können Sie endlich auf Ihrem Computer unter den überall herumliegenden Audiofiles Ordnung schaffen. Sie verwaltet und katalogisiert die Dateien auf sämtlichen Laufwerken.

Das wohl wichtigste Feature ist, dass in WaveLab die Bearbeitung in Echtzeit erfolgt (sogar bis zu sechs Prozesse gleichzeitig). Das heißt, wenn Sie einen Effekt zuweisen, können Sie diesen während der Wiedergabe an- und ausschalten. Das Programm verarbeitet Dateien bis zu 24 Bit und 96 kHz, und das auch im Stapelmodus. Die Audiomontage bietet unbegrenztes Undo/Redo. WaveLab ist ein offenes System und unterstützt neben der Direct-X-Technologie zahlreiche andere Plug-ins (VST 2.0).

Besonders gut gelungen ist das Wave-Bearbeitungsfenster. Es ist zweigeteilt, der untere größere Teil stellt einen Bereich vergrößert dar, der obere kleinere Teil zeigt immer die ganze Datei an. Auf diese Weise fällt die Orientierung auch bei größeren Sounds sehr leicht. Die Benutzeroberfläche und die Tastaturkommandos sind vom Benutzer programmierbar – jeder kann sie seiner individuellen Arbeitsweise anpassen. Man könnte die Hymne natürlich noch weitersingen ... aber lieber kurz gesagt: Wer ein wirkliches Profiwerkzeug braucht, der sollte WaveLab wählen.

Ich möchte hier noch ein sehr nützliches Tool vorstellen, und zwar den Multimedia Xplorer von Moonsoftware (Demoversion auf der CD unter Gestaltung/Sound/Programme). Wie der Name schon sagt, ist es ein dem Windows Explorer ähnliches Programm mit Zusatzfunktionen für multimediale Dateien. Die meisten Funktionen sind zwar für Bilddateien gedacht, lassen sich aber auch mit Sounddateien sehr gut verwalten. Mit einer Vorschaufunktion (die angeklickte Datei wird intern abgespielt) oder mit dem vordefinierbaren Favorite- und Destination-Ordner kann man Verschiebe- und Kopiervorgänge mit einem einzigen Mausklick erledigen. Sehr nützlich ist die Funktion Ausführen mit. Die meistgenutzten Programme lassen sich vorher definieren, um dann die Audiodatei mit einem Mausklick mit dem gewünschten Programm auszuführen. Mit dem Multimedia-Detektiv lassen sich die Festplatten nach weiteren Multimediadateien durchsuchen. Die Registrierung kostet $ 25.

Soundbearbeitung: Tipps

Ein scheinbar ganz einfach zu lösendes Problem ist es, die richtige Lautstärke zu treffen. Man neigt dazu, wenn ein Sound ganz leise ist, sofort zum Lautstärkeknopf zu greifen. Falsch!!!

In Profistudios werden die Lautstärkeknöpfe der Verstärker nie berührt. Die Apparate werden mit einem Messton kalibriert, um somit eine absolute Lautstärkeeinstellung zu schaffen. Bei Rock- und Poptiteln versucht man dann bei der Aussteuerung der Titel fast bis zur Verzerrungsgrenze zu gehen. Natürlich ist das bei Ihren Flash-Sounds nicht das Ziel. Es muss nicht alles bis in den roten Bereich gehen. Ein sanfter Mouseover-Sound muss nicht laut sein. Wie schaffen wir also eine mehr oder weniger verlässliche Umgebung?

Nun gehen wir davon aus, dass der bekannte Microsoft-Sound überall ertönt, wenn ein Windows-Rechner hochgefahren wird. Wenn er zu laut ist, dreht der Benutzer den Lautstärkeknopf bis in den angenehmen Be-

Abbildung 4 ▸
Die Kunst, einen Loop genau zu extrahieren

reich herunter, oder wenn er zu leise ist, dann umgekehrt. Machen Sie doch dasselbe! Stellen Sie die Lautstärke Ihres Verstärkers so ein, dass Sie den Microsoft-Sound (C:\Windows\Media\) als angenehm empfinden. Und ab jetzt Hände weg vom Lautstärkeknopf! Nutzen Sie die Lautstärkedynamik einzelner Sounds. Softsounds dürfen durchaus leise sein. Oft gemachte Fehler sind, dass die Sounds in einem Wave-Bearbeitungsprogramm »normalisiert« werden. Der stärkste Pegel wird gesucht – maximiert, und alle anderen Teile des Sounds dementsprechend angepasst. Damit erreicht man zwar eine einheitliche (und laute) Lautstärke, was aber überhaupt nicht immer erwünscht ist. Dynamik ist einer der wichtigsten Bestandteile der Musik, spielen Sie einfach damit!

Der wohl am meisten durchgeführte Arbeitsschritt ist die Erzeugung von Loops. Hier gilt folgende Faustregel: Wenn Sie von einem Musikstück einen Loop zuschneiden möchten, sollten Sie ein längeres Stück brauchbarer Musik zu Ihrem Loop machen. Es muss vorne und hinten noch verbleibendes Material vorhanden sein.

Machen Sie also nicht den Fehler, dass Sie bei der Aufnahme ungefähr am Anfangs des Loops starten und die Stopptaste circa am Ende des Loops betätigen. Jetzt ist also die Hauptaufgabe, den Start- und Endpunkt zu finden. Klicken Sie mit der Maus ungefähr dahin, wo der Startpunkt sein sollte. Meistens ist ein Drum-Schlag ein gut erkennbarer Pe-

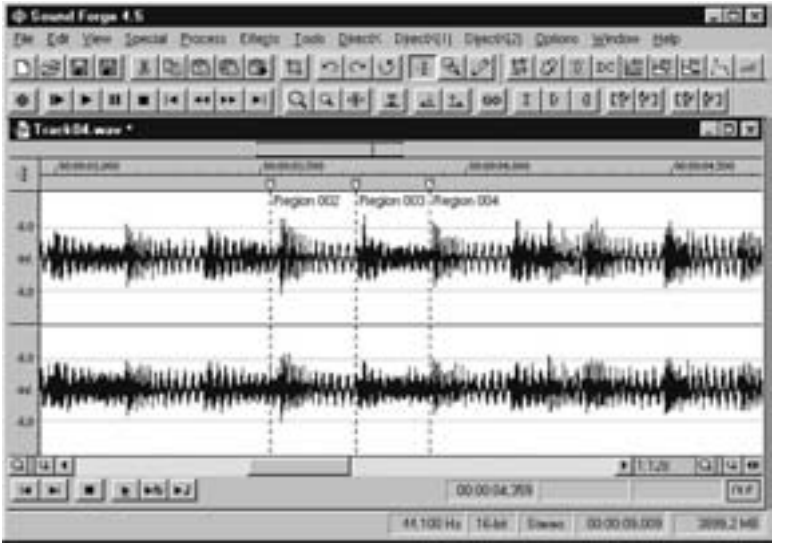
Sound Forge 4.5
File Edit View Special Process Effects Tools Direct X Direct X (1) Direct X (2) Options Window Help
Track04.wav *

◀ Abbildung 5
Loop einfügen

gel. Starten Sie die Wiedergabe ab diesem Punkt, und Sie werden sofort hören, ob der Punkt richtig sitzt oder nicht. Korrigieren Sie die Position nicht mit der Maus, sondern mit dem entsprechenden Tastaturbefehl. Man kann mit der Tastatur den Marker pixelweise verschieben (Sound Forge und WaveLab: die Pfeiltasten rechts und links). Eventuell setzen Sie auch die Lupe ein, um die zu bearbeitenden Bereiche besser erkennen zu können. Aber Vorsicht, bei zu großer Vergrößerung wird die Ansicht unübersichtlich, der Positionszeiger läuft zu schnell durch den Bildschirm. Wenn Sie schon sicher sind, dass der Punkt richtig sitzt, markieren Sie den Bereich ab dem Punkt bis zum Anfang, und zwar ebenso mit der Tastatur (⇧+Pos1), und löschen Sie den markierten Bereich (Entf). Kontrollieren Sie den Anfang noch einmal, sitzt er nicht korrekt, können Sie ihn jetzt nach einem Undo noch einmal markieren.

Nun suchen wir den Endpunkt. Sie sollten hierbei kein Ende suchen, sondern einen Anfang, an dem Ihr Loop noch einmal starten könnte! Ein oft gemachter Fehler ist es, dass man sich auf ein Ende konzentriert und den Endpunkt mit der Maus während der Wiedergabe festsetzt. Falsch!!! Sie sollten wie zuvor einen Startpunkt suchen – hier wird ja der Loop immer wieder neu gestartet. Verfahren Sie genau wie beim Startpunkt, nur dass Sie beim Auswählen die Tastaturkombination ⇧+Ende drücken. (Einen richtigen und einen falsch geschnittenen Loop finden Sie auf der

Abbildung 6 ▸
Autoregion von Sound Forge


Sound Forge 4.5
File Edit View Special Process Effects Tools DirectX DirectX(1) DirectX(2) Options Window Help
Region 002
Region 003
Region 004

CD unter Gestaltung/Sound/Beispiele/Loop01.wav bzw. Loop02.wav.) Sie können auch die Autoregions-Funktion verwenden, um Punkte zu finden. Dazu markieren Sie den Bereich, der den zu findenden Punkt beinhaltet, und nutzen Sie den Befehl AUTOREGION. Je nach Einstellung werden Sie nun mehrere Punkte vorfinden. Löschen Sie die überflüssigen, und positionieren Sie den Cursor mit den Tasten auf den Marker.

Eine weitere wichtige Funktion ist das so genannte Time-Stretching (bei Sound Forge: Time Compress/Expand; WaveLab: Zeitkorrektur). Sie kennen das Problem: Die Animation dauert zehn Sekunden, und Sie haben Ihren Wunschsound gefunden, aber der ist nur acht Sekunden lang. Kein Problem, Sounddateien lassen sich (natürlich innerhalb bestimmter Grenzen) ausdehnen oder zusammendrücken. Aber Vorsicht, Sie sollten behutsam vorgehen. Je nach Audiomaterial können unerwünschte Effekte auftreten. Wie viel Stretching Ihr Sound noch verkraften kann, müssen Sie ausprobieren. (Auf der CD Gestaltung/Sound/Beispiele finden Sie drei WAVs: 100 Prozent, 85 Prozent und 125 Prozent. Die zwei manipulierten Dateien stellen einen Grenzfall dar. Am Ende des Klangs treten schon unerwünschte Nebeneffekte auf.)

Musik selbst machen – MIDI & Co. – virtuelle und reale Studios

Ein Studio kann sich nicht jeder leisten. Anders verhält es sich hingegen mit virtuellen Studios. Dies sind Softwarelösungen, welche die Studiomaschinen vom Aufnahmegerät bis zum Effektprozessor und sogar Synthesizer simulieren. Diese erlauben, unter einer bestimmten Arbeitsoberfläche MIDI und Audio zu kombinieren. Und da tritt auch schon die Frage auf:

Was ist dieses MIDI überhaupt?

Wladimir Horowitz sagte einmal, dass Klavierspielen nicht schwer sei: Man müsse nur in dem richtigen Moment die richtige Taste anschlagen. Genau das macht MIDI! Es ist nichts anderes als eine Steuerungsdatei, die einen (oder mehrere) Synthesizer (Sampler, Expander etc.) steuert. Es weist den Synthesizer an, in welchem Moment im Zeitgeschehen welche Taste mit welcher Kraft geschlagen werden soll, so als hätte man eine Pianorolle gemacht. Sie kennen bestimmt diese Maschinenklaviere, die Titel von einer gelöcherten Pianorolle abspielen. Wenn man die Pianorolle herausnimmt und auf einem anderen Maschinenklavier abspielt, wird zwar derselbe Song ertönen, aber mit einer anderen Klavierklangfarbe. Wohl gemerkt geschieht die Klangerzeugung außerhalb der Datei durch ein Instrument. Deshalb sind MIDI-Dateien so klein.

Einen Tastenanschlag nennt man ein Event. Jedes Event hat bestimmte Eigenschaften wie den Zeitcode (zeigt die Entfernung vom Startpunkt des Songs), Tonhöhe, Anschlagdynamik etc. Ob das jetzt ein Klavierklang oder eine Trompete wird, wird mit einem zusätzlichen Befehl (Program Change) gesteuert. Die MIDI-Synthesizer sind in der Lage, gleichzeitig mehrere Instrumente zu simulieren. Nehmen wir an, wir nehmen eine kleine Melodie – gespielt auf einer Trompete – auf. Erst setzen wir den Program Change-Befehl, damit der Synthesizer die folgende Töne auf einer Trompete abspielt. Dazu sind die Instrumentensätze bei GENERAL (GM) MIDI standardisiert, es muss ja gewährleistet sein, dass – wenn z. B. ein Program Change-Befehl 57 gesendet wird – überall auf allen GM-MIDI-Synthesizern das Instrument Trompete angesteuert wird. Solche GM-Instrumentensätze sind auch in allen Soundkarten gespeichert.

Eine MIDI-Datei wird also auch ohne Hilfe von außen abgespielt. Nur wenn man die Qualität der Sounds verschiedener Hersteller hört, weiß man, dass zwischen Trompete und Trompete oder zwischen Klavier und Klavier riesige Unterschiede vorhanden sind. Die Instrumentensätze billiger Soundkarten sind schlicht und einfach unbrauchbar. Aber auch die

▲ Abbildung 7
Gold Platin

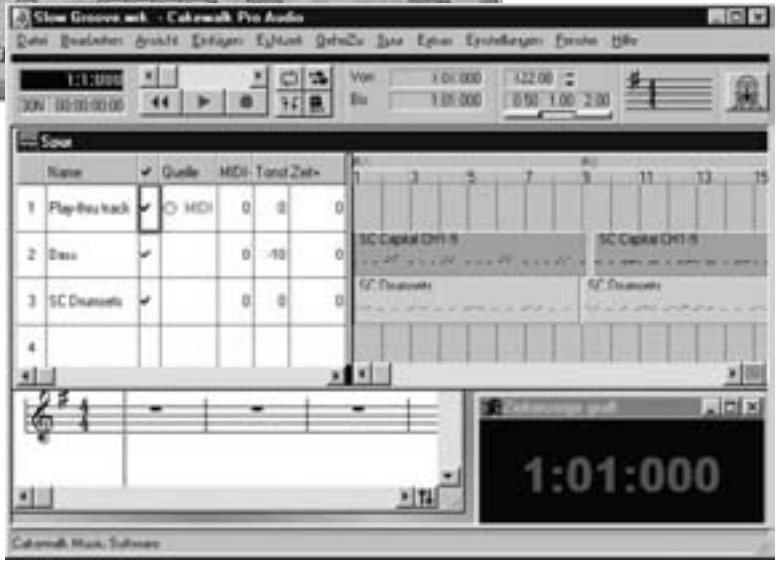
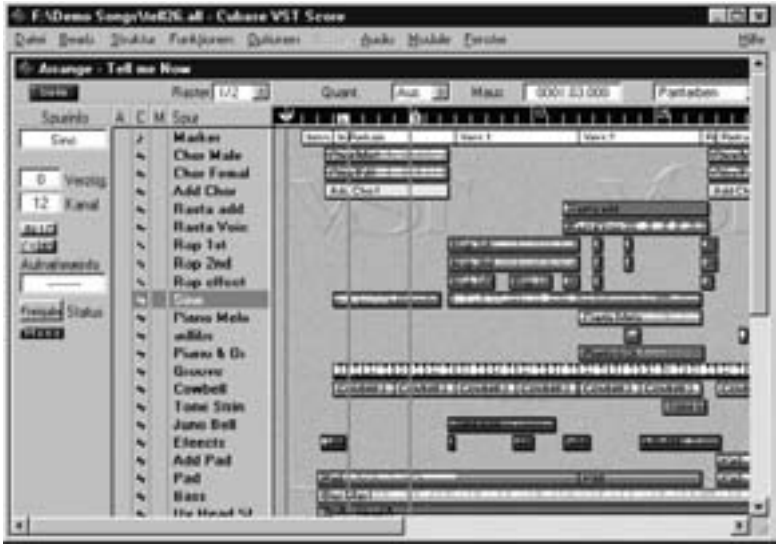


Abbildung 8 ▶
Cakewalk Pro Audio

Sounds teuerer Soundkarten erreichen die eindruckvollen, besseren Klänge der externen Expanderlösungen (wie z. B. Sound Canvas von Roland) nicht. Daher kann man nie sicher sein, wie eine MIDI-Datei klingen wird, wenn man sie dem Verbraucher überlässt.

Aber fahren wir nun mit unserer Aufnahme fort: Starten wir die Aufnahme genauso wie bei einer Audioaufnahme. Eine innere Uhr beginnt zu laufen, und es werden alle Tastenanschläge registriert. Fertig. Aber wir brauchen noch weitere Instrumente! Dazu müssen wir noch einen Begriff kennen lernen. Und das ist der MIDI-Channel. Jedem Instrument, das in unserem Song vorkommt, muss man auf dem Synthesizer einen Channel zuweisen. Unsere Trompete z. B. wird auf Channel 1 angesprochen, weitere Begleitinstrumente wie Bassgitarre und Klavier werden auf Channel 2 bzw. 3 gesteuert. Auf diese Weise kann man die Instrumente voneinander trennen und für Ordnung sorgen. Noch dazu ähnelt es den mehrspu-



◂ **Abbildung 9**
Cubase VST

rigen Bandmaschinen. Da sind die verschiedenen Instrumente auch auf getrennten Spuren aufgenommen.

So kann man Schritt für Schritt, Instrument für Instrument eine ganze Band aufbauen. Schön und gut, nur das schon erwähnte Problem: Wie klingt es schließlich beim User? Man sollte also die MIDI-Datei in eine Wave-Datei umwandeln. Es gibt so genannte Wave-Render-Programme, die diese Aufgabe übernehmen und ganz brauchbare Ergebnisse liefern können. (Auf der CD unter Sound/Programme finden Sie eines.) Die bessere Variante bleibt aber, das MIDI-File mit einem hochwertigen Klangerzeuger laufen zu lassen und das Ergebnis einfach als .WAV aufzunehmen.

Wer sich mit der MIDI-Problematik auseinander setzen will, muss zugegebenermaßen noch einiges dazulernen (unter anderem auch die Kleinigkeit des Klavierspielens).

Die drei bekanntesten Vertreter von virtuellen Studios sind:

1. Cakewalk Pro Audio
2. Cubase VST
3. Logic-Audio

Für Anfänger passt vielleicht das Cakewalk Pro Audio (DM 698) am besten, die Bedienung folgt der Windows-Systematik und ist also auch intuitiv zu erlernen. Die beiden deutschen Programme sind ähnlich in ihrer Leistungsfähigkeit, obwohl die Gemüter – wer welches liebt – hier sehr auseinander gehen. Sowohl Logic-Audio als auch Cubase VST sind in mehreren

Versionen erhältlich, die sich in Leistungsfähigkeit und Preis voneinander unterscheiden (DM 200 bis 1500). Die Programme sind in der Lage, bis zu 128 Audio- und unbegrenzt viele MIDI-Spuren abzuspielen (je nach Version). Im Prinzip ähneln sie den Mehrspur-Bandmaschinen: 1. Spur Drums, 2. Spur Bass, 3. Spur Gitarre etc. Die einzelnen Spuren lassen sich editieren:

Man kann Sie schneiden, kopieren, verschieben usw., aber natürlich sind auch WAV-Editor-Funktionen vorhanden. Die MIDI- und Audiospuren lassen sich zeitlich synchronisieren. Beim Mixdown hilft ein sehr leistungsfähiges virtuelles Mischpult.

Alles inklusive (?)

Einen weiteren Schritt in Richtung Professionalität bieten die Portable Home Studios. Vor ca. 20 Jahren fing die Firma Fostex an – neben anderen – kleine kompakte »Studios« auf dem Markt zu etablieren. Ein Portastudio vereint in sich ein Aufnahmegerät sowie ein Mischpult, eventuell auch mit Effektprozessoren, Kompressor und Noise Gate in einem einzigen preiswerten (?) Gerät. Die ersten Modelle konnten gerade einmal vier Spuren abspielen, als Datenträger diente die inzwischen fast ausgestorbene Audio-Compact-Kassette mit ihrer grausigen Klangqualität.

Die heutigen Modelle verdienen es schon eher, als Studio bezeichnet zu werden. Sie erreichen die Spurenzahl von 18 physikalischen Spuren und machen die Aufnahmen auf Festplatte oder ZIP-Disketten. Sicher sind sie nicht gerade billig, die Preisspanne liegt bei ca. 2000 bis 8000 DM, bieten aber den rechnerbasierten virtuellen Studios gegenüber einige Vorteile. Die Schalter, Fader und Transporttasten sind echt, werden nicht mit der Maus bedient, das System stürzt nie ab und lässt sich problemlos dahin bewegen, wo die Musik gespielt wird.

Tja, aber welche soll ich denn nehmen? Das Problem ist, dass wegen der Philosophie »All in One« die Geräte nicht oder nur sehr begrenzt erweiterbar sind. Die Entscheidung muss also »sitzen«. Nur wenn Sie mehrere Synthesizers oder andere Klangerzeuger zu Hause haben, lohnt sich ein Modell mit vielen Eingängen. Oder wenn Sie Liveaufnahmen planen, müssen Sie zu Geräten mit mindestens acht Eingängen greifen.

Über die Spurenzahl ist da noch einiges zu klären: Es existieren nämlich drei Arten von Spuren:

- physikalische
- gleichzeitig aufnehmbare
- virtuelle

Nehmen wir ein Beispiel: Der Roland VS-840GX hat acht physikalische, vier gleichzeitig aufnehmbare und 128 virtuelle Spuren. Das bedeutet, dass das Gerät in der Lage ist, gleichzeitig acht Spuren abzuspielen, aber nur vier Spuren gleichzeitig aufzunehmen. Jede der acht physikalischen Spuren ist weiter gespalten in 16 virtuelle Spuren (8 x 16 = 128). Es lässt sich aber immer nur eine von diesen virtuellen Spuren wiedergeben. Sie können also z. B. alternative Aufnahmen enthalten. Nehmen wir an, Sie nehmen Gesang auf. Sie können also einen zweiten, dritten usw. Versuch auf die gleiche physikalische Spur aufnehmen, später bei dem Mixdown müssen Sie sich aber für einen entscheiden. Mit Hilfe dieser virtuellen Spuren können Sie während der Arbeit mehrmals Submixes erstellen und somit sehr komplexe Aufnahmen realisieren.

Die Portastudios haben natürlich auch Effekte wie Hall, Delay, Chorus sowie die Dynamiksteuerungsprozessoren wie Kompressor oder Noise Gate. Ein absolutes Muss ist die Synchronisierbarkeit mit MIDI-Sequenzerprogrammen, und das am besten in beiden Richtungen.

Es gibt auf dem Markt eine breite Auswahl solcher Kompaktstudios. Nehmen wir stellvertretend von beiden Enden der Stange ein Exemplar.

Der Roland VS-840GX kostet DM 2890. Er verfügt über acht physikalische Spuren, für die gleichzeitige Aufnahme stehen vier Spuren zu Verfügung. Die Aufnahme wird auf eine 250-MB-ZIP-Diskette gemacht, und zwar mit einer Sampling-Rate von 44,1 kHz und mit der Auflösung von 20 Bit. Er speichert zwar in einem eigenen Format, die Dateien lassen sich aber als WAV exportieren.

Eine wesentlich höhere Ebene stellt das Modell Yamaha AW4416 dar. Es ist wirklich ein komplett ausgestattetes Studio mit 44 Kanälen, acht Gruppen und 26 Eingängen. Selbstverständlich gehört auch eine Multieffekt-Sektion dazu. Bedienen lässt sich das Gerät sogar mit der Maus. Zwar wird die Aufnahme (gleichzeitig 16 Spuren) auf eine Festplatte gemacht, aber für das Premastering ist auch schon gesorgt. Es ist ein Platz für einen CD-Brenner vorgesehen, und damit wird das Gerät wirklich »All in One«. Der stolze Preis von DM 7990 erscheint zwar recht hoch, ist aber durchaus berechtigt.

Unsere stark komprimierte Reise durch die wunderbare Soundwelt ist nun fast zu Ende. Wie Sie sehen, verlangt sie von uns einiges an Wissen, und manchmal sind auch finanzielle Opfer angesagt, aber es macht unheimlich viel Spaß, etwas Neues zu schaffen. Aber wem sage ich das, Sie sind ja Flasher ...

SoundObject

▶ Jan Schlünzen

So, jetzt kommen wir zur Praxis. In Flash gibt es eine interessante Neuerung: Das SoundObject. Mithilfe des SoundObjects lassen sich für Flashverhältnisse relativ komplexe Soundeffekte erstellen.

Im Folgenden bieten wir eine grundlegende Einführung in das SoundObject anhand einer kleinen Jukebox. Am Ende werden Sie eine Oberfläche erstellt haben, bei der der User zwischen drei Sounds wählen, diese abspielen und stoppen und außerdem auch noch die Lautstärke und das Soundpanorama (Verteilung des Sounds von links nach rechts im Stereo-Feld) selbst bestimmen kann.

Grundlegendes zum SoundObject

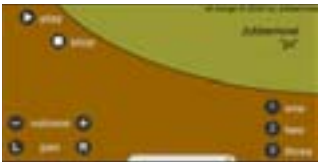
Bevor einem Sound überhaupt so genannte Methoden (z. B. Lautstärke, Stop, Play etc.) zugewiesen werden können, muss ein neues SoundObject erstellt werden. Diesem können wir dann wiederum einen Sound aus der Library zuweisen. Erst jetzt ist der Sound innerhalb von ActionScript verfügbar.

Vorbereitungen

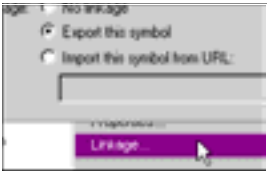
Als Erstes muss der Sound in die Flash-Umgebung importiert werden. In Flash 5 können nun auch MP3-Dateien importiert werden, Sie können aber auch weiterhin .wav- oder .aiff-Dateien einlesen. Für dieses Beispiel habe ich MP3-Files benutzt, um die Größe der FLA klein zu halten. Unter File·Import... (Ctrl+R) lässt sich ein Sound importieren. Importieren Sie drei gewünschte Sounds, die Sie für Ihre Jukebox benutzen möchten.

Wie schon am Anfang erwähnt, muss ein SoundObject erstellt werden. Um aber die Sound-Datei überhaupt einem Objekt zuweisen zu können, muss das Sound-File für die Flash-Umgebung **sichtbar** gemacht werden.

Öffnen Sie die Library (Ctrl+L): Hier finden Sie nun alle eben importierten Sounds wieder. Klicken Sie mit der rechten Maustaste auf einen der Sounds in der Library und wählen den Punkt Linkage... aus dem Kontextmenü. In dem jetzt folgenden Dialog muss Export this Symbol ausgewählt werden. Zusätzlich müssen Sie der Datei noch einen bisher nicht verwendeten Namen (den so genannten Identifier) geben, damit der Sound auch von ActionScript aus angesprochen werden kann. Wie-



▲ **Abbildung 10**
Die Jukebox



▲ **Abbildung 11**
Linkage

derholen Sie den letzten Schritt für alle Sounds in Ihrem Flash-Movie (achten Sie darauf, dass jeder Identifier nur ein Mal verwendet werden darf!). In meinem Beispiel heißen die Sounds »sound01«, »sound02« und »sound03«.

Am Anfang des Films muss nun ein neues Flash-SoundObject erzeugt werden, und einer der Sounds muss diesem zugewiesen werden: Erstellen Sie eine neue Ebene und nennen sie »actions«. Wählen Sie den ersten Keyframe in dieser Ebene aus und öffnen das ActionScript-Fenster (Ctrl+Alt+A). Schreiben Sie den folgenden Code:

```
song = new Sound();
song.attachSound("sound01");
song.setVolume(100);
song.setPan(0);
songname = "\"joi\"";
```

Hierdurch wird eine neue Variable `song` vom Typ Sound erstellt. Als nächsten Schritt wird die Sound-Datei »sound01« der Variablen zugewiesen (`song.attachSound("sound01")`). Hierbei handelt es sich übrigens nicht um den Namen der Sound-Datei, den man in der Library vorfindet, sondern um den Identifier, der vorher unter LINKAGE für den Export definiert wurde. Somit lässt sich der Sound jetzt auf einfache Weise ansprechen.

Als Nächstes werden für den Sound Startwerte initialisiert (eine Lautstärke von 100 % [`setVolume(100)`] und eine Zentrierung des Sounds auf beide Lautsprecher [`setPan(0)`] – mehr darüber später in diesem Kapitel). Zusätzlich habe ich für die Jukebox noch eine weitere Variable deklariert, die den Namen des Songs speichert, um ihn später auf der Oberfläche der Jukebox anzuzeigen. Etwas Grundsätzliches zur Syntax von Action-

Script: song. muss vor allen Methoden stehen, da Flash sonst nicht weiß, welchem Objekt er die neuen Einstellungen zuweisen soll.

Die Buttons – Einleitung

Nun kommen wir zum Erstellen der Buttons. Ich habe zu diesem Zweck vorgefertigte Buttons aus den »Common Libraries« benutzt und diese nur leicht verändert: ein Button für Play, Stop, Minus, Plus, links und rechts sowie die Buttons »one«, »two« und »three« zur Auswahl zwischen den verschiedenen Songs. Diese können Sie nun nach Belieben auf Ihrer Arbeitsfläche platzieren.

Die Buttons – Funktionalität

Damit das Ganze nun auch funktioniert, müssen den Buttons Aktionen zugewiesen werden.

Selektieren Sie den Play-Button und öffnen das ActionScript-Fenster. Tippen Sie folgenden Code ein:

```
on (release, keyPress "<Space>") {
   song.stop();
   song.start();
}
```

Wenn nun auf diesen Button geklickt wird (oder optional die Leertaste gedrückt wird), wird der Song abgespielt (`song.play()`). Das `song.stop()` sorgt dafür, dass der zuvor abgespielte Sound vorher gestoppt wird, damit kein Sound-durcheinander entsteht.

Selektieren Sie als nächstes den **Stop-Button** und fügen Folgendes in das ActionScript-Fenster ein:

```
on (release, keyPress "0") {
   song.stop();
}
```

Bei Klick auf den Button (oder Taste 0 auf der Tastatur) werden alle momentan laufenden Sounds gestoppt (`song.stop()`). Würde man nur einen bestimmten Sound anhalten wollen (da z. B. mehrere Sounds gleich-

zeitig ablaufen), müsste man noch zusätzlich in Klammern explizit den Sound angeben, der angehalten werden soll.

Etwas Grundlegendes zur **Lautstärke:** Die Skala reicht von 0 bis 100, wobei 0 für 0 % Lautstärke – also keinen Ton – und 100 für 100 % Lautstärke des Originalsounds steht. 100 ist die Voreinstellung, mit der ohne weitere Angaben alle Sounds abgespielt werden. Theoretisch sind auch Werte oberhalb 100 und unterhalb 0 möglich, die aber zu unerwünschten Ergebnissen führen: Unter null wird die Lautstärke wieder erhöht und oberhalb von hundert entstehen Verzerrungen. Mehr dazu im Folgenden.

Selektieren Sie den Minus-Button für die Kontrolle der Lautstärke und tippen Sie Folgendes ins ActionScript-Fenster:

```
on (release, keyPress "-") {
   if ((song.getVolume())>0) {
      song.setVolume((song.getVolume())-10);
   }
}
```

Bei Klick (oder der Minus-Taste) wird die Lautstärke des gerade aktuellen Sounds um 10 % herabgesetzt. Dazu wird mit Hilfe von `getVolume` erst einmal die aktuelle Lautstärke bestimmt, und von dieser werden dann 10 abgezogen (`song.setVolume((song.getVolume()) - 10)`). Die IF-Abfrage habe ich aus folgenden Gründen eingebaut: Das Volume in Flash kennt keine Vorzeichen, das heißt, wenn eine Lautstärke von null erreicht wird und der Benutzer noch einmal auf den Minus-Button klickt, wird das Volume nicht auf –10, sondern wieder auf 10 gesetzt, bei erneutem Klick auf 20 usw. Damit dieser unerwünschte Nebeneffekt nicht auftritt, habe ich eine Grenze von 0 eingebaut: Wenn eine Lautstärke von Null erreicht wird, werden die Aktionen für diesen Button nicht mehr ausgeführt.

Es folgt der Plus-Button für die Lautstärke:

```
on (press, keyPress "+") {
   if ((song.getVolume())<100) {
      song.setVolume((song.getVolume())+10);
   }
}
```

Gleiche Funktionsweise wie beim Minus-Button: Bei Klick (oder Taste +) wird die Lautstärke um 10 % erhöht. Hier hat die IF-Abfrage allerdings die Funktion, dass durch höhere Prozentzahlen als 100 %, die übrigens theoretisch auch möglich sind, Verzerrungen entstehen würden. Da das nicht unbedingt gewollt ist, habe ich also eine Grenze von 100 gesetzt. So kann eine Lautstärke nie auf einen Wert oberhalb von 100 steigen.

Die Buttons – Funktionalität (Fortsetzung)

Das Panorama legt fest, wie der Sound auf die beiden Lautsprecher verteilt ist, also wie viel Prozent des Sounds aus dem linken und wie viel aus dem rechten Lautsprecher kommen sollen. Die Skala reicht hier von –100 bis +100, wobei –100 für 100 % aus dem linken Lautsprecher und nichts aus dem rechten, 0 für ein mittiges Panorama (50 % links und 50 % rechts) und +100 für 100 % aus dem rechten und nichts aus dem linken Lautsprecher steht. 0 – also eine Zentrierung im Stereofeld – ist hier die Voreinstellung.

Zum Button L, der das Panorama nach links verschieben soll (der Sound kommt so, je öfter auf den Knopf geklickt wird, immer mehr aus dem linken Lautsprecher):

```
on (release, keyPress "l") {
   song.setPan((song.getPan())-10);
}
on (keyPress "c") {
   song.setPan(0);
}
```

Bei Klick (oder Tastendruck l) wird das Panorama um 10 % weiter nach links verlagert. Als Erstes wird der momentane Stand des Panoramas abgefragt (`song.getPan()`) und von diesem Wert werden dann 10 abgezogen (`song.setPan((song.getPan()) -10))`).

Zusätzlich habe ich hier noch die Möglichkeit eingefügt, bei Tastendruck C (für center) das Panorama mittig zu setzen (`song.setPan(0)`).

Der Button R, um das Panorama nach rechts zu verschieben:

```
on (release, keyPress "r") {
   song.setPan((song.getPan()) + 10);
}
```

Bei Klick (oder Tastendruck r) wird das Panorama um 10 % weiter nach rechts verlagert. Das Ganze funktioniert genauso wie der L-Button.

Die Buttons »one«, »two« und »three« haben die Funktion, zwischen den drei zur Auswahl gestellten Songs zu wählen. Die Idee ist die, dass dem definierten SoundObject »song« ein anderer Sound zugewiesen wird. So können alle Befehle wie Play, Stop oder Panoramaeinstellungen auf jeden beliebigen Sound angewandt werden, wenn dieser gerade dem SoundObject `song` zugewiesen ist.

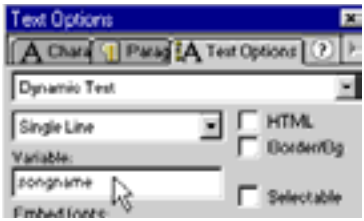
Funktionsweise am Beispiel des »one«-Buttons:

```
on (release, keyPress "1") {
song.stop();
song.attachSound("sound01");
songname = "\"joi\"";
song.start();
}
```

Bei Klick (oder Tastendruck »1«) wird der vorher spielende Song angehalten (`song.stop()`) und dem SoundObject `song` per `attachSound` ein neuer Sound zugewiesen (hier: `"sound01"`). Als Nächstes wird der Variablen `songname` der Name des aktuellen Songs zugewiesen. Dieser soll später in der Anzeige der Jukebox zu sehen sein. (Falls Sie die ungewöhnliche Syntax verwirren sollte: Das »\"« dient nur der Anzeige der Anführungszeichen im Flash-Movie. Würde der Schrägstrich nicht vor ein Anführungszeichen gesetzt werden, würde Flash dies als Beendigung des Variablennamens interpretieren. Der würde dann also nur »\« lauten. Deshalb also die Schrägstriche vor Anführungszeichen. Zum Schluss wird durch `song.start()` der Sound abgespielt.

Nach dem gleichen Prinzip funktionieren die weiteren Buttons »two« und »three«, nur dass diesen ein anderer Sound und ein anderer Songname zugewiesen werden muss.

Abbildung 12 ▶
Variable im Textfeld setzen



Der Songtitel

Im letzten Schritt muss nun nur noch der Songtitel eingebaut werden, der sich dynamisch verändert, je nachdem, welcher Song gerade der aktuelle ist.

Ziehe einen Textrahmen auf und wähle unter TEXT OPTIONS den Eintrag DYNAMIC TEXT. Damit Werte in dieses Textfeld hinein geschrieben werden können, muss dem Feld ein Variablenname zugewiesen werden. Hier kommt jetzt endlich die Variable `songname` zum Einsatz. Geben Sie dem Textfeld also diesen Variablennamen. Nun wird automatisch der jeweils aktuelle Songname in die Textbox geschrieben.

Das war's! Nun sind Sie in der Lage, mit dem SoundObject in Flash 5 umzugehen! Es gibt zwar noch einige zusätzliche Funktionen, zum Beispiel das Einbauen von Loops, ein Offset, so dass ein Sound erst ab einer bestimmten Stelle abgespielt werden soll, oder das so genannte »Transform« für weitergehende Effekte des Panoramas, aber das würde jetzt zu weit gehen. In der Flash 5-Hilfe sind noch einige Informationen zu diesen Themen zu finden.

3D

Die 3. Dimension ist nur mit Tricks zu erreichen

Die dritte Dimension ist im interaktiven Screendesign ein recht vernachlässigter Bereich – gerade in Flash! So viele Möglichkeiten Flash uns sonst auch bieten mag – mit der dritten Dimension tut sich das Webanimationstool Nr. 1 leider schwer.

Probleme bei der Erstellung von 3D

▶ Jan Frischmut

Die Schwierigkeiten liegen hauptsächlich in folgenden zwei Faktoren begründet:

Flash besitzt keine z-Achse

Als reines 2D-Animationstool besitzt Flash von Haus aus nur die x-und eine y-Achse – wirkliche Dreidimensionalität kann innerhalb von Flash also kaum erzeugt werden. Komplexe dreidimensionale Daten können nicht live berechnet werden und müssen als Animationssequenz Keyframe für Keyframe importiert werden – das kostet Kilobytes!

Wer eine längere Animation produzieren möchte und dabei auf kleine Dateigrößen angewiesen ist, wird viel Geduld aufbringen müssen, um markante Punkte und Linien in seiner Animation zu suchen und zu shapen.

Eine andere Möglichkeit wäre, sich eine kleine 3D-Engine in Flash zu schreiben – leider nicht ganz einfach und nur mit wirklich simplen Objekten realisierbar (siehe auch den Workshop 3D ab Seite 278).

Grafikkarten sind zu langsam

Was? Unsere hochgezüchteten Grafikbeschleuniger mit 64 MB DRAM und 400 MHz Takt und Ganzbild-Antialiasing sind zu langsam für das bisschen 3D?

Ja! Das Problem liegt darin, dass Flash keinen Zugriff auf beschleunigende 3D-Chipsätze hat, sondern sich ausschließlich auf den 2D-Chip der Grafikkarte verlässt (ist es doch ein 2D-Vektoranimationstool).

Leider sind auch bei den neuesten Grafikkarten genau diese Chips Modelle aus der goldenen 486er-Pentium-Ära. Zu dieser Zeit begann der 3D-Beschleunigerboom und so wurde das Hauptaugenmerk fast ausschließlich auf die Entwicklung noch schnellerer 3D-Chips gerichtet.

Eine bildschirmfüllende Einzelbildanimation bei 30 Frames mit anspruchsvollen Farbverläufen ist in Flash fast unmöglich zu realisieren – der Grafikchip, der das leistet, muss erst noch entwickelt werden. Die einzige Hoffnung ist, dass Flash es lernt, eines Tages die Ressourcen des 3D-Chips für zweidimensionale Darstellung zu nutzen. Allerdings bieten einige Hersteller auch heute noch speziell für den 2D-Bereich optimierte Chipsätze an – doch leider besitzen 98 Prozent der User die langsamen Standardgrafikkarten.

Die Konsequenz? Versucht man beispielsweise, 40 komplexe Vektorobjekte wie Asteroiden für einen Weltraumflug zu animieren, wird die als rasanter Flug gedachte Animation schnell zur gemütlichen Diashow – trotz 800er-Pentium III. Es hilft dann nur, für die Asteroiden Bitmaps zu verwenden. (Beispiel: www.powerflasher.de/elsa/intro – Flash 3D an der Leistungsgrenze moderner Grafikkarten)

Eine Möglichkeit, schon im Vorfeld festzustellen, ob ein Enduser ausreichend Hardware-Power besitzt, um sich eine besonders aufwändige Flash-Animation mit mindestens 20 dropout-freien Frames anzusehen, ist eine Speeddetection; mehr dazu im Kapitel »Speeddetection« ab Seite 607.

Erstellungsmöglichkeiten

Ein PC-Monitor liefert im Endeffekt nur zweidimensionale Bilder – räumliches Sehen ist nur mit Hilfe von LCD-Shuttle-Brillen oder anderen technischen Finessen möglich. 3D-Grafiken sind also eigentlich auch platt – sie können die Dimensionalität des Bildschirmes nicht verlassen.

Aber was macht dann 3D aus? – Perspektive!

Perspektivische Animationen geben dem Auge zumindest die Illusion eines räumlichen Objekts – es entsteht ein Verständnis für die Dreidimensionalität des gezeigten Gegenstands. Die simpelsten Beispiele hierfür sind z. B. Würfel, Pyramiden oder Kugeln. Diese Objekte sind einfach zu erzeugen, für eine Kugel reicht z. B. schon ein kreisförmiger Farbverlauf.

Ein Flasher, der sich nach dreidimensionalem Content sehnt, hat drei Möglichkeiten, mehr oder weniger aufwändige perspektivische Effekte zu erzielen.

1. die mathematische 3D-Erstellung
2. die 3D-Erstellung mithilfe von so genannten Pseudos
3. die 3D-Erstellung durch Sequenzen

Mathematische 3D-Erstellung

Dreidimensionale Bilder basieren zu 100 % auf mathematischen Berechnungen. Ein 3D-Renderer ist daher auch ein Vektorprogramm – nur dass er meist Pixeldaten ausgibt. Simple perspektivische Objekte lassen sich mit einigem Aufwand auch in Flash auf mathematischem Wege generieren.

Wie das? Trigonometrie!

Sites wie www.yugop.com oder www.derbauer.de zeigen z. B. solche winzigen 3D-Engines. Diese Engines sind ein kleines Stück wirkliches 3D in zweidimensionaler Flash-Umgebung.

Unter »Workshops: 3D Clipping« ab Seite 278 zeigen wir, wie solch eine 3D-Engine aussehen kann.

3D mittels »Pseudos«

Man presse einen Kreis zu einer dünnen, vertikalen Linie zusammen, lasse ihn wieder auf Originalgröße wachsen und ... erhält eine Münze!

Dies ist ein typisches Beispiel für einen simplen Pseudoeffekt. Zugegeben, solch eine Münze wirkt nicht wirklich plastisch, und ein auf diese Weise rotierter Text ist auch kein Eyecatcher.

Aber das Prinzip an sich kann durchaus auch effektvoll genutzt werden.

Als Beispiel sehen Sie die Abbildungen 1 bis 4: Dieser simple Pseudo-3D-Effekt kann für metallische Interfaces oder als Ticker genutzt werden.

◂ **Abbildung 1**
Hier unterstützen einige Farbverläufe die Illusion eines plastischen Objektes

◂ **Abbildung 2**
Die Schrift und das LED liegen in einem höheren Layer und werden darin sechs Frames lang gestaucht.

◂ **Abbildung 3**
Weitere sechs Frames lang ist dann nur unser »Rohr« zu sehen ...

◂ **Abbildung 4**
... bis dann Schrift und LED an der Unterseite wieder auftauchen usw. usw. – Loop!

3D durch Sequenzen

Die wirklich anspruchvollen 3D-Animationen entstehen jedoch aus Einzelsequenzen. Diese Sequenzen können eigentlich jeden beliebigen Effekt darstellen – als Bitmap-Sequenz sogar Hollywood-reife Partikeleffekte und andere 3D-FX-Tricks.

Der Nachteil: Sequenzen sind speicherintensiv und für den 2D-Flasher nicht einfach zu erstellen – der Umgang mit einem 3D-Renderer will gelernt sein!

Das Werkzeug

Wer sich bisher mit der Erstellung und Modellierung dreidimensionaler Objekte noch nicht beschäftigt hat, das aber nun tun möchte, sollte bei der Auswahl seines Werkzeuges sorgsam vorgehen. Nicht zuletzt da eine Lizenz für einen 3D-Renderer von DM 400 bis DM 36.000 und mehr kosten kann.

Ein wichtiges Kriterium ist, dass das Programm eine Exportfunktion in das allgemeine *.3DS-Format besitzt – wir benötigen dieses Format, um später unsere Objekte z.B. in Stand-alone-Tools in Vektoren umrechnen zu lassen.

Ein gutes Einsteigerprogramm, das auch durchaus von Profis noch genutzt werden kann, ist **Bryce** 4 von Metacreations. Bryce ist aber für Umweltdarstellungen ausgerichtet und erschwert dem User das Modellieren eigener Objekte ab einer gewissen Komplexität stark. Allerdings sind die Texturen leicht zu handhaben, so dass sich schnelle Ergebnisse erzielen lassen.

Der Allrounder und Klassiker bleibt weiterhin **3D Studio Max**. Eine deutsche Lizenz kostet zwar um die DM 8000, aber nicht zuletzt wegen der für Max erhältlichen Vektor-Plug-ins wie Vecta 3D (auch Stand-alone) und Illustrate kann sich der Kauf lohnen. Mit Max ist quasi alles machbar – »Jurassic Park« oder »The Matrix« belegen dies eindrucksvoll.

Modellierung

Wer Objekte modelliert, die später in Vektoren umgewandelt werden sollen, muss beachten, dass z. B. ein Zylinder in einem polygonalen Renderer kein wirklich rundes Objekt ist, sondern aus einzelnen Dreiecken besteht – so auch eine Kugel oder ein Kegel. Wirklich runde Objekte gibt es damit nicht – ab einer gewissen Anzahl Einzeldreiecke entsteht dieser Eindruck jedoch. Wenn Bitmap-Sequenzen erzeugt werden sollen, ist dies sinnvoll – für Vektorobjekte jedoch nicht! Jede zusätzliche Linie kostet mehr Byte, und ob eine Kugel aus 40 oder 120 Ecken besteht, fällt dem Betrachter nicht mehr auf.

Funktionen wie »MeshSmooth« sind damit absolut tabu – wo man kann, sollten Details vermieden werden. Man sollte auf die minimale Anzahl Polygone optimieren. Zu beachten ist auch, dass sich Stand-alones wie Swift 3D mit booleschen Objekten schwer tun – das Objekt wird einfach ignoriert und nicht berechnet.

Texturing

Wird ein Objekt für den späteren Vektorexport entworfen, ist es wichtig, immer im Hinterkopf zu behalten, dass außer einer Lichtreflexion in Form eines Farbverlaufes keine Texturierung möglich ist.

Der Eindruck einer Oberfläche mit starkem Bumpmapping – also starken Erhöhungen – kann in einigen Fällen durch Splines (Linien, die beim Modelling verwendet werden) oder ganz kleine Einzelobjekte erzeugt werden.

Export

Für den Export in einen Stand-alone-Vektorisierer muss nur beachtet werden, dass das *.3DS-Format gewählt wird. Die Bewegungen werden mit exportiert und können z. B. bei Swift 3D später noch angepasst werden.

Wird eine Bitmap-Animation exportiert, ist es wichtig, dass dafür das *.BMP-Format gewählt wird. Bitmaps komprimiert Flash generell besser als JPEGs, und gerade bei Bitmap-Sequenzen ist jedes Byte kostbar.

Beispiel: www.powerflasher.de/operationx – 800 KB großer Trailer für ein Online-Game.

Vom 3D-Modell zum Vektorbild

Es gibt drei Wege, um vom 3D-Modell zur Vektoranimation zu kommen:

- Handtracing
- 2D-Vektorisierung
- 3D-Vektorisierung

Handtracing

Der bis vor kurzem noch einzige Weg vom 3D-Modell zum vektorisierten Objekt war das Handtracing. Den ersten wirklich konsequenten Einsatz fand die Technik auf der ausgezeichneten Site von Manuel Clement (www.manone.com), der den stark beeindruckten Flashusern in einem Tutorial erstmals das Handtracing erklärte.

Der Vorgang ist so einfach wie zeitaufwändig. Das zu animierende Objekt wird ohne Texturen in einem handelsüblichen 3D-Renderer erstellt. Es muss nicht immer gleich Lightwave oder 3DS Max sein, Simply 3D z. B. reicht völlig aus.

Zu beachten ist nur, dass das Programm eine Animation nicht nur als AVI, sondern auch als Bildsequenz exportieren kann, denn dies wollen/müssen wir hier ausnutzen. Die Sequenz sollte möglichst klein gehalten werden, obwohl das weniger eine Frage der Dateigröße als eine Frage der Geduld ist. Denn Handtracing ist eigentlich nicht mehr als ein bloßes Nachzeichnen von Konturen.

Die einzelnen Bilder sind zuerst vorzugsweise als BMP – oder als ein anderes Pixelformat, das Flash importieren kann – abzuspeichern. Die

entstandenen Dateien sind unbedingt von dem ersten Bild der Sequenz an aufsteigend durchzunummerieren. Das übernimmt meist der Renderer, notfalls muss man es selbst erledigen. Nun kann man das erste Bild der Sequenz in Flash importieren, Flash erkennt das Bemühen, eine Bildsequenz zu importieren, und erspart das stundenlange Anlegen von Keyframes und immer neues Importieren der einzelnen Bilder.

Es kommt also ein Abfrage, ob man die Bildsequenz importieren möchte. Der Rest ist schnell erklärt: Den Layer mit den importierten Bildern schützen, dann einen neuen Layer erstellen. Dabei darauf achten, dass er sich über dem Bilderlayer befindet. Wie man die Zeichenfunktionen in Flash benutzt, muss ich ja keinem mehr erklären – los geht's! Zugegeben ... diese Lösung eignet sich nur für Geduldige mit reichlich Vektorerfahrungen.

Wer es sich nicht leisten will, 40 % bis sogar 80 % an Dateigröße zu verschenken, der sollte auch bei »maschinell« gerenderten Animationen selbst Hand anlegen – es lohnt sich!

Was uns Stand-alones und Plug-ins vektorisieren, können Sie nur in Keyframes ablegen – die Vektorversion einer Bitmap-Sequenz ...

Aber was, wenn im Laufe einer Animation auf eine Kugel gezoomt wird? Erst einmal besteht eine Kugel, egal wie viele Segmente sie besitzt, aus einzelnen Polygonen und besitzt somit nie eine völlig runde Oberfläche – dies liegt in der Natur des Polygonsystems. Mathematisch anders ausgerichtete Renderer wie »Organic« z. B. erzeugen ausschließlich Kugeln an Stelle von Dreiecken. Vecta, Swift und Co. rechnen also für eine Kugel dementsprechend viele einzelne Linien zusammen. Es entsteht eine Menge Datenmüll (= Trash). Dabei wäre ein simpler Kreis viel kleiner – Flash kann ihn auf mathematischem Wege errechnen. Die Position, Größe und Gruppierung einzelner Pixel ist nicht fest gespeichert, sondern nur als mathematische Variable vorhanden. Die Vorteile kennen wir: freie Skalierbarkeit und geringere Dateigröße.

Es bietet sich also an, das Zoomen auf eine Kugel auf mathematischem Wege zu lösen, ohne Linien, ohne Keyframes (na ja – zwei müssen sein). Fazit: Ein simples Shapen kann viel wirkungsvoller sein als zehn gerenderte Frames. Die Zoom-Animation der Kugel oder des Kreises über 40 Einzelbilder wäre 40 mal so groß wie ein Tweening.

Viele Formen in einer 3D-Animation können so behandelt werden – solange sie nicht so komplex sind, dass sie sich beim Shapen zu neuen Zielpunkten orientieren und so völlig andere Formen als gewünscht ergeben.

Hier können Formmarken helfen – aber auch nur in 80 % aller Fälle. Vorher wird eben dieser zu shapende Teil der Animation aus den Keyframes der restlichen Animation gelöst und in einen anderen Layer gebracht – das kostet Zeit und Nerven, die Mühe lohnt sich aber. Damit sind gute kleine und schnelle 3D-Vektoranimationen eine reine Zeit-, Geld- und Präzisionsfrage.

Das oben angesprochene Problem, dass sich in gerenderten Animationen keine wirklich runden Linien, sondern nur viele Einzelteile einer Rundung befinden, ist einer der größten Speicherfresser. Was kann man tun, wenn nicht getweent werden kann? Per Hand Keyframe für Keyframe Kreise einzeichnen? Ich erwähne das hier, um Experimentierfreudigen herbe Enttäuschungen zu ersparen – es geht leider nicht.

Die Ungenauigkeit, die das Augenmaß leider aufweist, führt über die ganze Animation gesehen dazu, dass das so nachgezeichnete Objekt unschön zittert, es sitzt in jedem Keyframe einige Pixel versetzt. Eine stillstehende Kugel könnte auf diese Weise zwar noch behandelt werden, aber sobald sie sich bewegt, treten Probleme auf; wir haben leider nicht die Möglichkeit, mathematisch genau zu positionieren.

Besonders Vecta 3D neigt dazu, Linien übereinander zu legen – FreeHand perfektioniert diese unschöne Eigenart geradezu. Ein in FreeHand getracetes Bild wird zuweilen in sechs Ebenen übereinander gruppiert – eine wahre Speicherfalle! Es lohnt sich also, nicht nur auf eventuell zu shapende Formen zu achten, sondern stichprobenartig zu überprüfen, ob Linien doppelt gerendert wurden, ob wirklich jede Linie für die Perspektive wichtig ist und ob eventuell sogar Linien übereinander gruppiert worden.

2D-Vektorisierung

2D-Vektorisierung bedeutet nichts anderes, als dass ein Bitmap auf markante Linien, Formen und Farben untersucht wird und diese als Vektoren getraced werden.

Flash bietet von Haus aus diese »Trace Bitmap«-Funktion. Es kann voreingestellt werden, ob Flash eher harte oder weiche Kurven finden soll, wie groß ein Farbbereich mindestens sein muss, um als Vektor erkannt zu werden, und wie viele Ecken erkannt werden sollen.

Prinzipiell gilt: Je weniger Linien, desto besser – dennoch gilt für jedes Bitmap eine eigene Einstellung: man muss dies austesten!

▲ Abbildung 5
Oben: Bitmap, unten: Trace

Der Mindestwert von drei Pixeln sollte jedoch nie unterschritten werden – das Tracing würde sonst garantiert größer als die Bitmap-Vorlage.

FreeHand bietet ebenfalls eine Tracefunktion, genannt AUTOTRACE. Das im Vergleich zu Flashs TRACE BITMAP umfangreichere Tool gruppiert einmal gefundene Objekte übereinander. Das muss nicht immer praktisch sein – gerade bei Farbverläufen werden Gruppen schnell zum Problem.

Allerdings können so einzelne Objekte gezielt separiert und animiert werden – ein Vorteil. Zudem ist FreeHand in der Lage, vorher festzulegen, wie viele Einzelfarben erkannt werden sollen.

Ob nun Flash oder FreeHand – generell gilt: Handtracing nach dem Autotracing; es können leicht einige Kilobyte gespart werden, wenn man manche unnötigen Gruppen entfernt.

Das Programm Streamline aus dem Hause Adobe ist sogar in der Lage, Linien als komplette Pfade zu erkennen und komplette Bitmap-Sequenzen direkt in Vektoren umzuwandeln. In Sachen Tracing sicher das umfangreichste Tool – Tracen ist aber auch seine einzige Aufgabe.

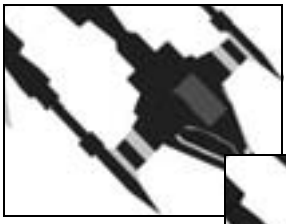
3D-Vektorisierung

... war etwas, von dem Flasher mit 3D-Ambitionen lange Zeit nur träumen konnten. Doch mittlerweile haben sich immerhin drei Tools als brauchbar erwiesen, und wenn sie auch keine perfekten Ergebnisse liefern, so sind sie doch alle recht brauchbar. Es sei vorweggenommen, dass sich alle drei Programme/Plug-ins ergänzen und ganz eigene Stärken und Schwächen aufweisen. Die Komplettlösung gibt es nicht zu kaufen, es sei denn, man schafft alle drei an und benutzt sie in Kombination.

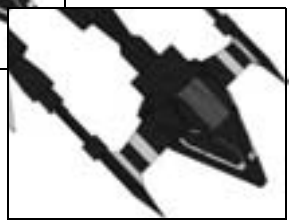
Ob als Plug-in oder Stand-alone – alle diese Tools gleichen der Render-Engine eines 3D-Renderers bis ins Detail, nur dass sie nicht erst Vektoren in Pixel umrechnen, sondern direkt Vektoren errechnen und in das *.SWF-Format exportieren.

Es muss unterschieden werden, ob das Tool nur als Plug-in für einen bestimmten Renderer oder als eigenständiges Programm universell eingesetzt werden kann. Das Tool Illustrate ist ausschließlich als Plug-in für 3D Studio Max erhältlich. Vecta 3D ist ein Plug-in für Max und Lightwav, aber auch als eigenständiges Stand-alone-Tool. Swift 3D ist ein kompletter Renderer, der sogar nachträgliche Objektanimation/Positionierung erlaubt. Alle diese Tools bieten nur die Möglichkeit, Linien und Farben zu rendern – Texturen sind nicht exportierbar.

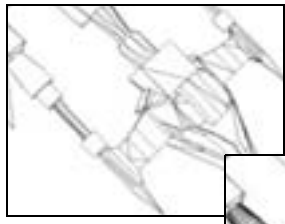
Es werden folgende Exportfunktionen unterschieden:



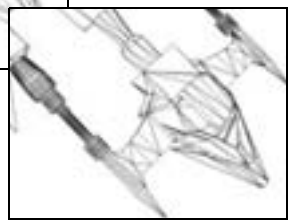
▲ Abbildung 6
Raumschiff mit der Option Filled



▲ Abbildung 7
Filled with Lines



▲ Abbildung 8
Outlines



▲ Abbildung 9
Mesh

Renderstyles

- Filled
  Die Option FILLED zeichnet die Oberflächen eines Polygonobjektes mit Farbe gefüllt auf den Monitor. Illustrate kann dabei allerdings nur einfarbig arbeiten – Farbverläufe kennt es nicht, wohl aber Schatten.
- Filled with lines
  Siehe FILLED; diese Option rendert lediglich die »Outlines« zu den Farbflächen. Also alle Linien, die nicht durch andere Objekte verdeckt werden, die Außenkanten eben.
- Outlines
  Als Outlines werden alle Linien, die an den Kanten der zu rendernden Objekte zu sehen sind, bezeichnet. Linien, die dabei durch Objekte verdeckt werden, sind darin nicht eingeschlossen. Es entsteht ein Umrissmodell, das einer Skizze ähnelt.
- Mesh
  Als Mesh-Rendering bezeichne ich einen Ausgabemodus, der zusätzlich zu den Outlines noch alle eigentlich verdeckten Linien hinzurendert. Illustrate und Vecta verstehen darunter nur die Außenkanten der Polygone, Swift alle Linien eines Dreiecks. Illustrate berücksichtigt beim Meshmode oder Wireframemode als einziges Programm die Arbeitsfarben der Objekte und rendert die Linien dementsprechend farbig.

Swift 3D ist bisher das einzige Programm, das in der Lage ist, die Funktion FILLED sogar mit Farbverläufen zu rendern – ein absolutes Plus!

Vor- und Nachteile der einzelnen Tools

Illustrate

Das Programm Illustrate gibt es seit etwa 1996. Kurz bevor 3D Studio Max, das ja damals noch 3D Studio R4 hieß, den Sprung auf die Windows-Plattform schaffte, erschien das Plug-in erstmalig. Illustrate renderte damals wie heute Polygonobjekte, ohne deren Texturen zu berücksichtigen. Lediglich Farben und Linien werden berechnet. Die Besonderheit ist, dass Illustrate mit dem »Rastermode« zum Rendern von comicartigen Animationen oder Bildern verwendet wird. Das Programm kommt mit einer für Rendering-Plug-ins nicht ungewöhnlichen Optionsvielfalt daher und bietet z. B. die Möglichkeit, nur die äußeren Linien (Outlines) oder die Linien aller Polygone (Mesh) zu rendern. Hierdurch findet es auch in Bauzeichnung und Planung Verwendung.

Ende 1998 erschien mit der Illustrate 4.0-Beta ein wahr gewordener Traum. Flasher mit 3D-Ambitionen konnten ihre Objekte und Animationen aus 3D Max mithilfe dieses neuen Plug-ins in das Vektorformat *.AI (Adobe Illustrator) oder gar direkt in eine importierbare SWF-Datei schreiben lassen. Illustrate befindet sich als Demo auf der Seite des Autors – www.davidgould.com. In der Version 4.0 (3D Studio Max R2) und 4.1 (3D Studio Max R3) kann eine Demo geladen werden.

Das Programm schreibt sich als Plug-in in die Menüleiste von 3D Studio Max und ist fortan neben dem Menüpunkt Hilfe zu finden. Die Anordnung des Menüpunkts ist durchaus bezeichnend. Zeigt sich bei der ersten Konfiguration des Render-Vorgangs noch ein praktischer Wizard, so wird der User bei einem zweiten Klick auf den Menüeintrag von einer regelrechten Optionsflut erschlagen.

Der letzte Ausweg ist da meist nur, sich nochmals des Wizards zu bedienen, um z. B. eine simple Einstellung versuchsweise zu ändern. Erschwerend kommt hinzu, dass nur wenige Optionen wirklich sinnvoll sind bzw. vom Flasher benötigt werden.

Ruft der User den Wizard auf, muss er in vordefinierter Reihenfolge das Ausgabeformat, die Farbe des Render-Hintergrunds, den Renderstyle (siehe oben), die Berücksichtigung von Linien und/oder Flächen und die Ausgabeauflösung wählen – die dank Skalierbarkeit für uns ja unerheblich ist. Einige Optionen können später auch noch in den Renderoptionen von 3D Studio Max angepasst werden.

Nicht allein auf Grund seines Alters werde ich an dieser Stelle Illustrate hart kritisieren müssen, daher nehme ich die positiven Punkte vorweg:

Illustrate rendert schnell und recht logisch, es gruppiert die einzelnen Objekte in der Ausgabe, wenn diese sich dann meist auch überschneiden oder unzusammenhängend gruppiert wurden. Im Vergleich zu den anderen Tools ist dies jedoch ein echtes Plus. Illustrate besitzt als einziges Programm einen Meshmode, der auch wirklich verwendbar ist. Im Gegensatz zu Swift 3D beschränkt es sich dabei auf markante Linien; Vecta 3D besitzt nicht einmal einen Meshmode und scheidet hier aus der Wertung aus. Eine mit Illustrate gerenderte SWF enthält wenig Trash – es gibt kaum doppelte Linien und wenige Überschneidungen, auf jeden Fall spart es KB. Im Fazit rendert Illustrate sogar von allen Programmen/Plug-ins am sparsamsten. Vecta erzielt teilweise die fünffache Dateigröße und Swift oft mehr als die doppelte ... Jedoch:

Illustrate ist komplex in der Detailbedienung – somit eine echte Herausforderung für einen Flasher, der im Bereich 3D erste Gehversuche macht. Flasher, die Erfahrung mit 3D Studio Max haben und an Render-Plug-ins gewöhnt sind, fühlen sich jedoch sofort heimisch.

Der Mode FILLED von Illustrate ist das größte Manko des Plug-ins. Flackernde Flächen, die teilweise ihre Farbe wechseln oder auf einmal nicht mehr gefüllt sind, treten in jedem Render-Vorgang auf – Handtracing ist angesagt! Linien, die z. B. an den Triebwerken eines Raumschiffs außer Sichtweite sein sollten und überhaupt nicht gerendert werden müssten, tauchen auf einmal über dem Cockpit auf ...

Zusammenfassend bleibt zu sagen, dass Illustrate für jemanden, der auf Farbverläufe verzichten kann, 3D Max besitzt und es schafft, das Tool zu beziehen und zu bedienen, das effizienteste Plug-in ist, das man derzeit bekommen kann: wenig Trash, kleine Dateigröße, Mesh-Rendering.

Beispiele zu mit Illustrate gerenderten Flash-Szenen findet man unter http://www.davidgould.com. Einige sind recht eindrucksvoll.

Vecta 3D

Anfang 1999 wurde Vecta 3D erstmals angekündigt. Heute ist die Plug-in-Version für 3D Studio Max recht weit verbreitet – eine Stand-alone-Version für den PC existiert bereits und wurde mit dem letzten Release von Flash 4 als Demo vertrieben.

Für den Mac ist ebenfalls eine Stand-alone-Version geplant – auf der Website der Entwickler www.ideaworks.com können sich Macianer per Mail benachrichtigen lassen, sobald diese lieferbar ist. Samples und Demos aller Versionen sind unter www.vecta3d.com zu finden.

▲ Abbildung 10
Der Hubschrauber hat eher einen platten Charakter

▲ Abbildung 11
Ein Fisch mit Vecta 3D

Vecta erweist sich als einfach zu handhaben und integriert sich in 3D Studio Max sehr gut. Linienfarbe und -stärke können direkt jedem einzelnen Objekt zugewiesen werden. Praktisch! Zu erwähnen ist, dass Vecta sehr schnell rendert – schneller als Illustrate. Allerdings produziert es von allen Vektorrenderern am meisten Abfall und erfordert hinterher eine Menge Geduld beim Handtracen.

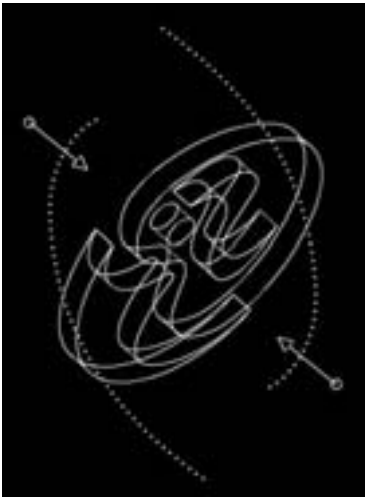
Die Stand-alone-Version soll – wenn sie denn offiziell erscheint – in der Lage sein, polygonale Schatten zu rendern. Das Ganze sieht dann nicht so schön aus wie in Swift 3D, sondern hat eher einen platten Charakter (siehe Abbildung 10).

Es bleibt zu sagen, dass Vecta ein für Einsteiger sehr einfach zu bedienendes Plug-in ist, sich dabei aber lange nicht so leistungsfähig zeigt wie Illustrate und technisch meilenweit hinter Swift 3D liegt – wir bleiben gespannt auf die Stand-alone-Versionen, die in der Demo schon einen guten Eindruck machen.

Swift 3D

Seit Anfang des Jahres 2000 ist 3D nun in aller Munde und kaum eine Flashsite ohne rotierenden Loading Button zu finden. Der Auslöser ist ganz klar Swift 3D von der Firma Electronic Rain, die normalerweise Plug-ins für 3D Max schreibt.

Swift ist ein Stand-alone-Tool, das schon allein dadurch auffällt, dass es auch für den Mac verfügbar ist. Vecta 3D z. B. ist derzeit den Mac-Usern noch vorbehalten (eine Stand-alone-Version für den Mac ist in Arbeit), und Illustrate ist absolut ein Plug-in für 3D Max. Unnötig zu erwähnen, dass 3D Studio Max auf den Apple-Plattformen nicht läuft.



◄ **Abbildung 12**
Rotierende Objekte gibt es seit dem Erscheinen von Swift 3D auf fast jeder Flash-Site

Swift macht es dem User einfach, per Wizard *.3DS-Dateien zu importieren, zu positionieren und zu animieren. Dabei versteht Swift sogar Splines und interpretiert komplexe Animationen, die in diesem Format abgelegt werden können. *.EPS-Dateien können ebenfalls eingeladen und extruiert werden. So ist der Flasher in der Lage, quasi jedes erdenkliche Logo in Swift zu importieren, zu animieren und als Vektorsequenz in Flash zu verwenden. Zudem ist Swift in der Lage, dreidimensionalen Text aus jeder Schriftart zu extruieren und primitiven zu erzeugen. So können sogar Laien schnell 3D-Textanimationen und einfache geometrische Objekte erzeugen.

Das Tool macht es dem User einfach, Lichtquellen hinzuzufügen, Glanzmaterialien aufzubringen, zu positionieren und zu animieren. Sogar häufig genutzte Animationsabläufe können per Drag and Drop einzelnen Objekten zugeteilt werden. Spätestens die Erwähnung von Lichtquellen oder Glanzmaterialien hätte stutzig machen müssen. Wir wissen ja bereits, dass Vektorrenderer generell nicht in der Lage sind, Texturen oder Farbverläufe darzustellen.

Und an dieser Stelle erhebt sich Swift meilenweit über seine Konkurrenten.

Der ganze Stolz von Electronic Rain ist die so genannte RAVIX 3D Vektor-Render-Engine, und das zu Recht. Die RAVIX Engine ist die Entwick-

Abbildung 13 ▸
Eine Maske mit Swift

lung, die das möglich macht, was Vecta 3D schon lange angekündigt hat: Vektorisierte 3D-Objekte mit echten Farbverläufen!

Swift berechnet seine Glanzmaterialien nach dem Lichteinfall der frei positionierbaren Lichtquellen. Zusätzlich zu dem Flatrenderer beherrscht Swift also noch das Area Shading. Das **Area Shading** kommt einer realen Renderausgabe ohne Texturen sehr nahe. Allerdings ist schon ein Frame einer Area Shading-Animation bis zu 40 KB groß.

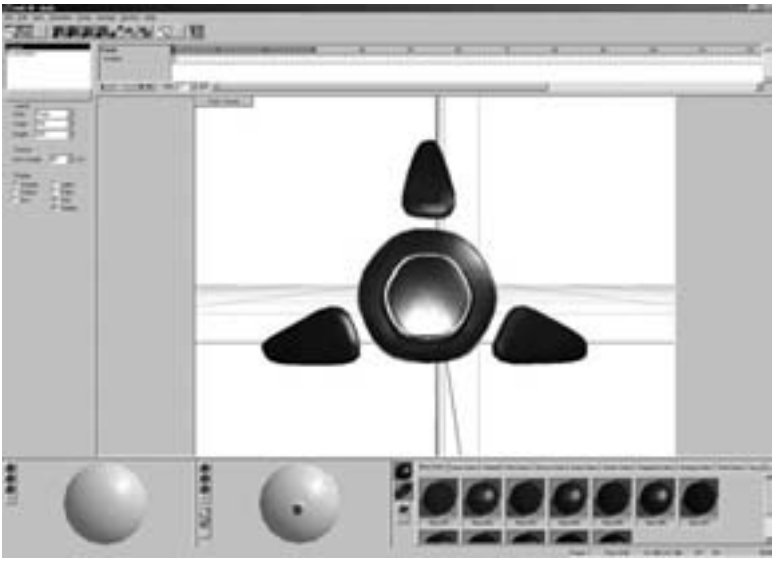
Eine Besonderheit weist ebenfalls der Meshmode von Swift auf: Er rendert jedes Polygon einzeln. Bei einigen Objekten sind das bis zu 3000 x 3 Linien. Der Rendermode ist also nicht allzu sinnvoll und nicht gerade arm an KB. Alle anderen Rendermodes leisten gute und relativ fehlerfreie Arbeit – fast so gut wie die von Illustrate. Allerdings gibt es auch in Swift wieder unschöne Flicker bei gefüllten Renderings – gerade beim Area Shading.

Beispiele, Demos und die Kommentare der begeisterten Fachpresse sind unter www.Swift3D.com zu finden.

Workshop 3D-Clipping

▸
Eric Singhartinger,
www.esreserve.com

Wie Sie bereits erfahren haben, fehlen uns in Flash zu echten Dreidimensionalität zwei entscheidende Dinge: Erstens ein Vektorraum, in dem wir arbeiten können, und zweitens eine Engine, welche die Projektion dieses Raums auf den Monitor übernimmt.



◀ **Abbildung 14**
Swift 3D –
Die Oberfläche

Der Vektorraum

Die Erstellung eines Vektorraums ist einfacher, als man denkt. 2D bietet Flash schon (x-Achse, y-Achse), wir benötigen also nur noch eine dritte Dimension für einen Raum (z-Achse). Und die denken wir uns zunächst einfach. Es gibt nämlich nur zwei Unterschiede, die ein nahes Objekt von einem entfernten Objekt unterscheiden: Größe (`_xscale, _yscale`) und Detail/Deutlichkeit (`_alpha`).

Die Projektion

Von einer Engine zu sprechen wäre bei diesem relativ einfachen Beispiel übertrieben. Viel muss die Engine nämlich nicht leisten.

Jedes Objekt hat einen X-, Y- und Z-Wert. Die beiden ersten übernehmen wir so, wie sie sind, und setzen unsere Objekte einfach an die jeweilige X- und Y-Position in unserem Movie, den Z-Wert benutzen wir etwas transformiert, um Größe (`_xscale, _yscale`) und Detail/Deutlichkeit (`_alpha`) zu bestimmen.

Das Beispiel

In diesem Beispiel rotieren mehrere verschiedenfarbige Bälle um einen unsichtbaren Mittelpunkt. Die gesamte Animation folgt dabei dem Mauszeiger in X- und Y-Richtung und lässt sich per Mausklick zusätzlich in die Z-Richtung bewegen. Die Rotation ändert sich dabei stets in Richtung des Mauszeigers.

Für unser Beispiel benötigen wir folgendes:

- Zunächst ein paar Bälle, die wir bewegen wollen. Ich nenne sie `blue`, `green`, `red` und `yellow`.
- Des Weiteren benötigen wir je einen MovieClip `ball`, `animation`, `library`, `mouse` und `script`.
- Zuletzt einen Button namens `klickobserver`, dessen Inhalt und Funktion ich im Folgenden erkläre.

Der MovieClip ball

Der Aufbau ist relativ einfach. Jedes Frame enthält einen der vier farbigen Bälle. Der erste Frame beinhaltet zusätzlich ein `Stop`, damit der MovieClip nicht bei seinem ersten Aufruf automatisch abgespielt wird. In die einzelnen Frames kann man theoretisch jedes beliebige Objekt einfügen, egal ob Grafik, Button oder MovieClip. Je komplexer das Objekt ist, desto langsamer wird jedoch die Animation. Zusätzlich sollten die Grafiken nicht zu groß sein, damit keine Überschneidungen während der Rotation entstehen.

Der MovieClip animation

Dieser enthält alle sich bewegenden Objekte. Genauer gesagt acht Instanzen des Objekts `ball`, benannt `ball1`, `ball2` ... `ball8`. Die Position der Bälle ist dabei egal, da sie beim Start des Movies mit Hilfe des Skript `script` platziert werden.

Der MovieClip liegt in der Hauptebene `_level0` des Movies. Instanzname ist `a`.

Der MovieClip mouse

Dieser MovieClip enthält den Button `klickobserver` nebst Actions.

```
on (release) {
   _root.trigger = 1;
}
on (press) {
   _root.trigger = -1;
}
```

Der erste Handler setzt die Variable `trigger` in der Hauptebene `_level0` auf 1, der zweite auf –1. Die Funktion erkläre ich im Skript `script`.

Der MovieClip liegt in der Hauptebene `_level0` des Movies. Instanzname `mouse`.

Der Button klickobserver

Dieser Button ist unsichtbar und enthält deshalb nur im Hitbereich einen kleinen Kreis. Der Button liegt im MovieClip `mouse`.

Der MovieClip script

Und last but not least, das Skript `script`. Hier passiert sozusagen alles. Der MovieClip liegt in der Hauptebene `_level0` des Movies, Instanzname ist `script`.

Das Skript wurde soweit als möglich objektorientiert erstellt, wodurch Objekte leicht hinzugefügt (`duplicateMovieClip()`), entfernt (`removeMovieClip()`) oder verändert werden können. Dieses Beispiel beschränkt sich jedoch auf die Rotation bereits vorhandener Objekte.

Das Skript unterteilt sich in zwei Bereiche. Der erste setzt die Umgebungsvariablen und positioniert die Objekte, der zweite Loop-Teil (`gotoAndPlay()`) ist für die kontinuierliche Rotation im Raum verantwortlich. Dieser Teil wertet zunächst die Zustandsvariablen (Mausposition, Maustaste, Rotation) aus und berechnet daraus die neuen Positionen der Animation und der einzelnen Bälle. Danach werden sie sortiert und neu gezeichnet.

Setzen der Umgebungsvariablen und Positionieren der Objekte

Der Drag des MovieClips `mouse` wird gestartet, damit die Position des Mauszeigers ermittelt werden kann.

```
_root.mouse.startDrag( true );
```

Die ersten beiden Variablen sind für die Bewegung in die Z-Richtung zuständig. `trigger` repräsentiert den aktuellen Sollwert in Z-Richtung, `depthZ` ist der aktuelle Istwert. `perspective` ist ein Wert für die Perspektive der Animation. Hohe Werte bedeuten keine Perspektive – niedrige Werte erhöhen die Perspektive.

```
// ___ creating environment
_root.trigger = 100;
depthZ = 100;
perspective = 120;
```

Die Variable `objects` bestimmt die Anzahl der zu rotierenden Objekte. Sie wird für die `for`-Schleifen im Loop-Teil des Skripts benutzt.

Diesen Part gibt es für jeden der acht Bälle, die zusammen einen Würfel bilden. Er initialisiert den X-, Y- und Z-Wert und zusätzlich die Farbe jedes einzelnen Balls. Zu beachten ist dabei, dass die jeweiligen Variablen in den einzelnen Bällen selbst sind! Die jeweiligen Werte können beliebig verändert werden.

```
//___ creating objects
objects = 8;
_root.a.ball1.X = -50;
_root.a.ball1.Y = -50;
_root.a.ball1.Z = -50;
_root.a.ball1.Type = 1;
```

Rotation im Raum

Zunächst wird die Tiefe der Animation `a` berechnet. Dabei erreicht die Variable `trigger` Werte zwischen –100 und 100. Sollte einer der beiden Extreme überschritten werden, so wird der Wert manuell in diesem Bereich gehalten. Der Wert wird beim Drücken (Press) der Maustaste auf –1 gesetzt und bei jedem Durchgang bis zum Wert –100 verdoppelt. Beim Loslassen (Release) wird der Wert auf 1 gesetzt und ebenso bei jedem Durchgang bis zum Wert 100 verdoppelt. Der Wert `trigger` wird für die Z-Bewegung der gesamten Animation verwendet. Sie setzt die Größe (`_xscale, _yscale`) und das Detail (`_alpha`) auf einen Wert zwischen `30` und `90`.

```
_root.trigger *= 2;
if (_root.trigger>100) {
   _root.trigger = 100;
} else if (_root.trigger<-100) {
   _root.trigger = -100;
}
```

Danach wird die Rotation, die sich aus der Position von Maus `mouse` und Animation `a` ergibt, berechnet.

Damit sich die Änderung der Rotation weicher vollzieht, lasse ich immer einen Teil der alten Rotation in die neue mit einfließen.

Obwohl solche mathematischen Operationen auf den ersten Blick etwas kompliziert aussehen, werde ich sie nicht näher erklären, da sie mit dem probeweisen Einsetzen von Testwerten leicht zu verstehen sind.

```
// ___ set rotation
newRotationX = (_root.mouse._y - _root.a._y) / 25;
newRotationY = -(_root.mouse._x - _root.a._x) / 25;
rotationX = newRotationX - (newRotationX - rotationX) / 1.1;
rotationY = newRotationY - (newRotationY - rotationY) / 1.1;
rotationZ = 0;
```

Der darauf folgende Teil ist der letzte, der sich mit der Speicherung und Auswertung des momentanen Zustandes befasst. Die Animation `a` wird einen Bruchteil näher in Richtung Maus `mouse` bewegt, ist also für die Bewegung der gesamten Animation in X- und Y-Richtung zuständig.

```
// ___ follow mouse
_root.a._x = _root.a._x+(_root.mouse._x-_root.a._x)/20;
_root.a._y = _root.a._y+(_root.mouse._y-_root.a._y)/20;
```

Gespeichert und ausgewertet dreht sich nun alles um die eigentliche Rotation. Da ich das Rad nicht neu erfinden will, verwende ich das Standardverfahren für Matrixrotation im dreidimensionalen Vektorraum. Dieses Verfahren bietet außerdem die Translation und Skalierung an, wovon ich aber hier keinen Gebrauch machen werde. Die Matrixrotation behandelt nacheinander die Drehung um die x-, y- und auch z-Achse. Die Drehung um die Z-Achse wird jedoch in diesem Beispiel außer Acht gelassen, kann aber ohne Probleme mit eingebaut werden.

Ich erkläre die Rotation um die x-Achse um den `winkelX` repräsentativ für die beiden anderen Rotationen aus der Sicht von Flash.

Y = Y x cos (winkelX) + Z x sin (winkelX) und
Z = Y x –sin (winkelX) + Z x cos (winkelX)

sind die beiden Transformationen für die x-Achse.

Für die Bestimmung des neuen temporären (!) Y-Wertes hole ich mir den aktuellen Y-Wert aus dem entsprechenden Ball und multipliziere ihn mit dem Wert, der sich aus der Variable ergibt, die ich aus dem String `/library/:cos` und dem Rotationswert `rotationX` zusammenbaue. Dazu addiere ich das Produkt aus dem Z-Wert des entsprechenden Balls und multipliziere ihn analog mit beispielsweise `eval("/library/:sin15")`, was im MovieClip der Library `Set Variable: "sin15" = 0.258819045116126` entspricht.

Dies wird ersetzt durch den nächsten Block:

Y = Y x cos (winkelX) + Z x sin (winkelX) und
Z = Y x –sin (winkelX) + Z x cos (winkelX)

sind die beiden Transformationen für die x-Achse.

Für die Bestimmung des neuen temporären (!) Y-Wertes hole ich mir wieder den aktuellen Y-Wert aus dem entsprechenden Ball und multipliziere ihn mit dem Cosinus von `rotationX`.

Dazu addiere ich das Produkt aus dem Z-Wert des entsprechenden Balls und multipliziere ihn analog mit dem Sinus von `rotationX` . Die trigonometrischen Funktionen in Flash erwarten ihre Parameter in Radianten und nicht in Grad. Mit der folgenden Funktion wandeln wir Grad in Rad um:

```
function grad2Rad (Winkel) {
   return Math.PI/180 * Winkel;
}
```

Die Funktion `grad2Rad` erwartet einen Winkel in Grad und gibt den Winkel als Radiante zurück.

```
// ___ calculate each object
for ( i=1; i<=objects; i++ ) {
   _root.a["ball"+i].tempY = _root.a["ball"+i].Y
         * Math.cos(grad2Rad(rotationX))
         + _root.a["ball"+i].Z
         * Math.sin(grad2Rad(rotationX));
}
```

Rotation für die y-Achse:

X = X x cos (winkelY) + Z x –sin (winkelY),
Z = X x sin (winkelY) + Z x cos (winkelY)

Rotation für die Z-Achse:

X = X x cos (winkelZ) + Y x sin (winkelZ),
Y = X x –sin (winkelZ) + Y x cos (winkelZ)

Am einfachsten ist es, sich die Funktionen selbst herzuleiten. Anderer Leute Code ist im Allgemeinen sehr schwer zu lesen.

Z-Clipping mittels Bubble Sort

Das eigentlich Beste an diesem Code ist das funktonierende Z-Clipping. Unter Z-Clipping bezeichnet man üblicherweise das Überdecken von weiter hinten liegenden Objekten durch weiter vorne liegende. In Flash besitzt jedes Objekt eine bestimmte Höhe, ähnlich des Levels, den man bei Duplicate MovieClip oder beim LoadMovie-Befehl angeben muss, das heißt, es liegt über oder unter einem anderen Objekt bzw., dreidimensional gesehen, vor oder hinter einem Objekt.

Da man die Sortierreihenfolge von Objekten, das so genannte **Stacking**, nach dem Kompilieren des Films nicht mehr ändern kann, muss man sich eines Tricks behelfen: Nach der Kalkulation der neuen Werte jedes einzelnen Objekts und noch vor dem Zeichnen sortiert man blitzschnell alle Objekte den Z-Werten nach um. Dabei verwende ich das einfachste aller gängigen Sortierverfahren, den Bubble Sort. Doch wie funktioniert das Umsortieren? Bubble Sort überprüft, ob der Z-Wert des ersten Objekts größer ist als der des zweiten. Tritt dieser Fall ein, werden X-, Y- und Z-Werte der beiden Objekte vertauscht. Zusätzlich werden die Werte der Variable **Type** vertauscht, was bewirkt, dass ebenso die Art des Objekts (hier die Framenummer im Objekt ball) mit vertauscht wird. (Zum Verständnis kann man das Vertauschen des Types unterbinden, indem man die letzten drei Zeilen des Bubble Sorts probeweise löscht.) Danach wird obige Bedingung für die nächsten beiden Elemente überprüft und gegebenenfalls wieder umsortiert. Nach und nach rücken so die vorderen Objekte nach vorne, die hinteren nach hinten. In Wirklichkeit rücken aber gar nicht die Objekte selbst, sondern nur alle Properties (X, Y, Z, Type) an eine andere Position.

O.k., ein Beispiel zum leichteren Verständnis: Während des Karnevals will ein Fotograf drei besonders gelungene Kostüme (hier die Bälle) samt ihren Trägern (Level) fotografieren: Sie stellen sich hintereinander auf. Doch der Fotograf ist mit der Anordnung nicht zufrieden, deshalb sollen sich seine Modelle in einer anderen Reihenfolge (Bubble Sort) aufstellen. Dummerweise haben sich die drei Personen aber in ein frisch geteertes Stück Asphalt gestellt und kleben nun im Teer fest. (Nach Kompilieren kein Ändern des Stackings mehr möglich.) Was also tun, damit der Fotograf doch noch sein Foto bekommt? Die drei Modelle entledigen sich schnell ihres gesamten Kostüms (X, Y, Z, Type) und tauschen dies nach dem Willen des Fotografs (IF-Statements). Nun ist der Fotograf zufrieden und schießt sein Foto mit der neuen Anordnung der Kostüme.

```
// ___ bubblesort for clipping
for ( i=1; i<=objects; i++ ) {
 for ( k=1; k<=objects-i; k++ ) {
  if (_root.a["ball"+i].Z>_root.a["ball"+(i+1)].Z) {
   temp = _root.a["ball"+i].Z;
   _root.a["ball"+i].Z=_root.a["ball"+(i+1)].Z;
            _root.a["ball"+(i+1)].Z=temp;
   temp = _root.a["ball"+i].Y;
   _root.a["ball"+i].Y=_root.a["ball"+(i+1)].Y;
            _root.a["ball"+(i+1)].Y=temp;
   temp = _root.a["ball"+i].X;
   _root.a["ball"+i].X=_root.a["ball"+(i+1)].X;
            _root.a["ball"+(i+1)].X=temp;
   temp = _root.a["ball"+i].Type;
   _root.a["ball"+i].Type=_root.a["ball"+(i+1)].Type;
            _root.a["ball"+(i+1)].Type=temp;
  }
 }
}
```

Die Bewegung der gesamten Animation in Z-Richtung

Ziemlich wenig Code, möchte man meinen, muss man doch alle acht Objekte unabhängig von der Rotation in Z-Richtung verschieben. In Wirklichkeit ist es aber viel einfacher: Man ändert lediglich Größe (`_xscale, _yscale`) und Detail/Deutlichkeit (`_alpha`) der gesamten Animation `a`. Per `_root.a._xscale` lässt man sich die aktuelle Skalierung des MovieClips `a` ausgeben und speichert ihn in der Variablen `nowZ`. Mit Hilfe der bereits oben beschriebenen Variablen `trigger` und einer kleinen Transformation bestimmt man den Sollwert für die Skalierung. Danach berechnet man die derzeitige Geschwindigkeit der Animation `a` und schließlich die daraus resultierende neue Z-Position. Mit `depthZ`, dessen Wert ungefähr zwischen `10` und `110` liegen sollte, werden nun die drei Properties `_xscale, _yscale` und `_alpha` gesetzt. Mit Hilfe dieses Verfahrens kann man die Bewegungen beliebiger MovieClips »nachfedern« lassen.

```
// ___ depth
nowZ = _root.a._xscale;
targetZ = 0.3*_root.trigger+60;
speedZ = speedZ/1.2+(targetZ-nowZ)/10;
```

```
depthZ = depthZ+speedZ;
_root.a._yscale = depthZ;
_root.a._xscale = depthZ;
_root.a._alpha = depthZ;
```

Rendern des neuen Zustandes

Zu guter Letzt müssen wir die Szene aktualisieren, das heißt die neu berechneten X-, Y- und Z-Werte sowie das neue Z-Clipping zeichnen. Hierzu verwendet man eine Schleife, welche die jeweiligen Variablen ausliest und den zugehörigen Properties zuweist. Erwähnenswert hier ist einzig die View-Transformation. Diese gibt den X- und Y-Positionen und -Scales je nach Z-Wert extremere Werte. Von »keine Perspektive« (`perspective > 500`) bis »Fischauge« (`perspective = 120`).

```
// ___ paint each object
for ( i=1; i<=Objects; i++ ) {
   _root.a["ball"+i].gotoAndStop (_root.a["ball"+i].Type)
   view = _root.a["ball"+i].Z/perspective+1;
   _root.a["ball"+i]._x = _root.a["ball"+i].X*view;
   _root.a["ball"+i]._y = _root.a["ball"+i].Y*view;
   _root.a["ball"+i]._xscale = (_root.a["ball"+i].Z/2+200)*view;
   _root.a["ball"+i]._yscale = (_root.a["ball"+i].Z/2+200)*view;
   _root.a["ball"+i]._alpha = _root.a["ball"+i].Z/2+75;
}
```

Ausblick: Was geht?

Wie Sie gesehen haben, bleibt auch bei 3D die Verwendung von ActionScript nicht aus. Nun ist das noch ein sehr einfaches Beispiel von 3D in Flash. Zusätzlich denkbar wäre Funktionalität zum Hinzufügen und Entfernen beliebig vieler Objekte, Erweitern der Objektvielfalt oder sogar die Möglichkeit zu Erstellung eigener 3D-Wireframes. Wie so etwas aussehen könnte, kann man sich auf meiner Website (http://www.esreverse.com) ansehen. »3D WIRE« ist ein solches Projekt, das in wöchentlichen Abständen verbessert wird.

Vielen Dank auch an Hassan von den Powerflashern, der sich noch die Mühe gemacht hat, den Code nach Flash 5 zu portieren. Einen anderen Ansatz für solch eine Engine findet man übrigens unter www.powerflasher.de unter den Referenzen.

HowTo

290 Typografischer Sitecheck

294 Sitecheck

294 www.takito.com
297 www.die-waescherei.com
300 www.flashspiele.de
303 www.schech.net
307 www.toferer.at
311 Elsa – das Intro
315 SSX – Electronic Arts
316 www.montblanc.com

Typografischer Sitecheck

Exemplarisch wollen wir Ihnen hier zeigen, wie auf Flash-Seiten mit Typografie umgegangen wird. Von Michael Bundscherer.

FELLOWS: SCHMECKEN SO GUT, WIE SIE RIECHEN.
Die Fellows riechen besser, weil das Aroma des Tabaks und sein unverkennbarer Vanilleduft sich jetzt noch harmonischer entfalten.
Fellows neu entdecken...
Mit und ohne Filter.

Abbildung 1 ▶
www.fellows.de: Der Webauftritt von Fellows ist gelungen. Spannung erzeugt auf jeder Seite ein großes stimmungsvolles Bild, das dem Text gegenübergestellt ist. Hier wird zwar eine Serifenschrift verwendet, die aber wegen ihrer Größe gut lesbar ist und zum Markenimage passt.

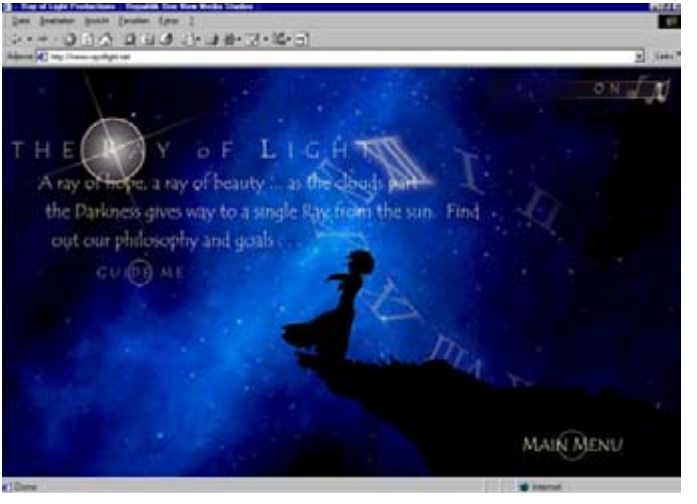

A ray of hope, a ray of beauty ... as the clouds part the Darkness gives way to a single Ray from the sun. Find out our philosophy and goals ...
MAIN MENU

Abbildung 2 ▶
www.rayoflight.net: The Ray of Light, ein Designunternehmen, verwendet auf seinen Seiten eine ziemlich eigenwillige, aber passende Schrift. Diese kann wegen des erhöhten Zeilenabstands und der auch bei niedrigeren Auflösungen ausreichenden Schriftgröße gut erkannt werden. An wenigen Stellen ist allerdings der Kontrast der Schrift zum Hintergrund zu gering.

megaCar
Willkommen
Der in das Auto integrierte Hochgeschwindigkeitszugang ins Internet ist endlich keine Vision mehr, er existiert!
Die mobilen Kommunikationslösungen der megaCar AG bieten neben Multimedia und Videoconferencing den Internetzugang im fahrenden Auto mit mehr als doppelter ISDN-Geschwindigkeit.

◂ Abbildung 3
www.megacar.com:
Auf klar gestalteten Seiten stellt megacar sein Konzept vor. Schon im Firmenlogo wurde auf Kontraste geachtet. Wegen der geringen Wortanzahl pro Zeile musste aber teilweise die Laufweite erhöht werden, um einen Blocksatz zu erreichen. Dadurch wirkt der Text unruhig.

◂ Abbildung 4
www.stefanboehme.net: Stefan Böhme ist Fotograf. Deshalb lässt er auf seiner Homepage vor allem seine Bilder »sprechen«. Trotzdem wird hier die Typografie nicht vernachlässigt. Die Versalien zum Beispiel sind gekonnt gesperrt. Bei längeren Texten (wie in der Abbildung) ist der Zeilenabstand leicht erhöht, und derText wird – wegen der kleinen Schriftgröße – ohne Antialiasing gezeigt. Es wurde einfach ein Screenshot des Textes aus einem Layoutprogramm als GIF importiert. Somit kann der Gestalter der Seiten auch das Aussehen des Textes optimal beeinflussen. Schade nur, dass der rechte Rand des Textes ziemlich stark flattert; Worttrennungen wären ja möglich gewesen. Die Buchstaben und Wortabstände sind leider nicht optimal.

Abbildung 5 ▶
www.rockstargames.com: Rockstargames hält auf seinen Seiten Informationen für den Besucher bereit. Damit ein Text auf einer Seite unterzubringen ist und diese trotzdem nicht überladen wirkt, hat man sich hier für einen scrollbaren Text mit einer zur Schriftgröße passenden Satzbreite entschieden. Wegen der relativ kleinen Schrift wurde die Schriftgröße nicht in den Film eingebunden, dadurch wird die Schrift nicht weichgezeichnet. Leider kann man deshalb aber den Zeilenfall nicht mehr sinnvoll beeinflussen.

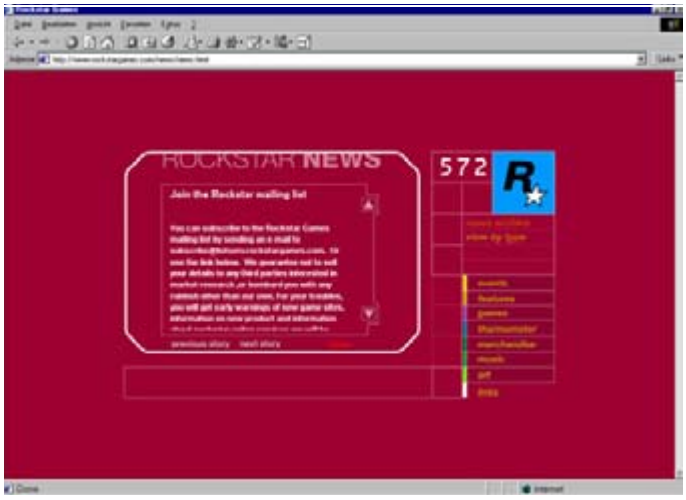
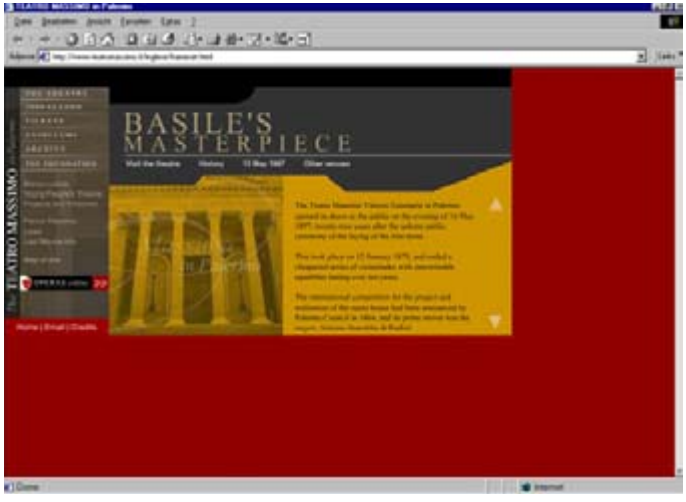
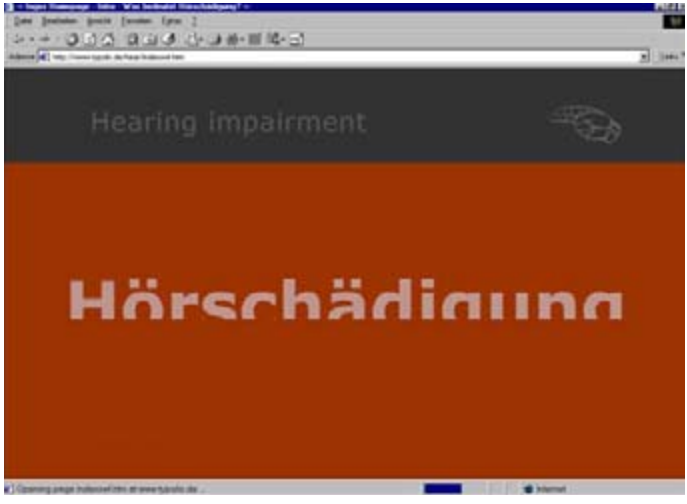


Abbildung 6 ▶
www.teatromassimo.it: Auch das Teatro Massimo in Palermo setzt auf seiner gelungenen Seite bei längeren Beiträgen auf scrollbare Texte. Die Textschrift ist, obwohl mit Serifen, noch gut lesbar. Gut, dass die Gestalter Versalien gesperrt haben, allerdings nicht generell und leider uneinheitlich.



◂ Abbildung 7
www.tobisstudiocanal.de:
Auf dieser Seite von Tobi wurde viel mit Kontrasten gearbeitet. Die Gestaltung wirkt dadurch sehr spannend. Negativ fallen hier nur die leicht unterschiedlichen Achsen des Haupttextes etwa zur Seitenüberschrift oder der Navigation und die etwas zu langen Zeilen auf.



◂ Abbildung 8
www.typolis.de/hear:
Diese Abbildung zeigt das Intro zu einer privaten Homepage zum Thema Hörschädigung. Da nur zum Thema hingeführt werden soll und die eigentlichen Inhalte erst später kommen, konnte hier auf umfangreiche Texte verzichtet werden. Stattdessen versucht man hier das Thema unter anderem typografisch zu interpretieren.

Sitecheck

Wie machen es die Profis?

In diesem Kapitel möchten wir Ihnen einige Seiten vorstellen und sie durchleuchten. Die Macher persönlich werden die Projekte, ihre Schwierigkeiten und die Lösungen beschreiben.

www.takito.com

▶ Gerald Marischka

Die Taki-To-Kindermoden-Seite ist eines der ungewöhnlichsten Projekte, das bisher durch meine Finger gelaufen ist. Das futuristische Design passt erst auf den dritten Blick zur eher konservativen Mode, und um die gewünschten Effekte zu erreichen, mussten wir viele Bilder einsetzen, was ja bekanntlich nicht gerade die Stärke von Flash ist. Diese Seite ist soweit entfernt von einer direkten Zielgruppe entwickelt, dass sie an sich nie ein Erfolg sein dürfte, oder doch?

Nach einigen Gesprächen mit dem Kunden und mithilfe der Kreativarbeit von Christian Pilsl entstand eine Seite, die es zu diesem Thema so nur im Internet geben kann. Selbstverständlich haben wir auf das CI Rücksicht genommen, aber die Seite von Taki-To hat – bis auf das Bildmaterial – so gut wie nichts mit der klassischen Werbung des Kunden gemeinsam.

Ziel war es, eine Seite für Internetbenutzer zu machen, die Stil hat, ungewöhnlich ist und die momentanen Kompatibilitätsvoraussetzungen nicht überfordert.

Nachdem doch beträchtlich viele Bilder, Sound- und Animationseffekte eingearbeitet wurden, entstand ein 17 MB großes FLA, das mit beträchtlichem Aufwand auf ein insgesamt nur noch 800 KB schweres SWF exportiert wurde.

800 KB sind nach heutigen Gegebenheiten eindeutig noch immer zuviel, um sie in einem einzigen Preload zu laden. Es gibt zwar Seiten im Netz, die ihre Besucher mit solchen »mörderischen« Preloads auf die Folter spannen, für uns ist das allerdings sicher keine Lösung. Um die wirkliche Ladezeit vor dem Öffnen der Seite möglichst gering zu halten und

◀ **Abbildung 1**
Die Taki-To-HomePage

dennoch nicht x verschiedene Preloads quer durch die Seite zu setzen, teilte ich die Seite in zwei Hauptgruppen:

- die Hauptseite, also Navigation, Inhalte und Sounds (ca. 200 KB)
- die Galerie mit dem Bildmaterial (ca. 600 KB)

Die Hauptseite wird dabei mit einem ganz normalen Preload geladen, der über `ifFrameLoaded` verschiedene Ladezustände an den Besucher als Feedback zurückgibt.

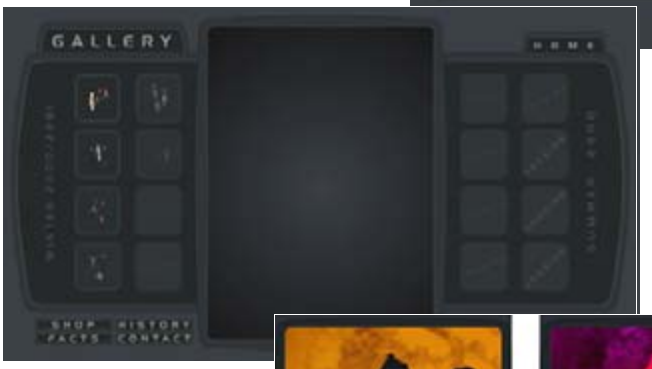
Durch eine kleine Animation neben dem Ladezustand sorgen wir zusätzlich dafür, dass die Ladezeit nicht langweilig wird.

Ist die Hauptseite geladen, so wird sie automatisch geöffnet, und der Besucher kann ab jetzt Informationen sammeln. Er merkt dabei nicht, dass im Hintergrund bereits jedes einzelne Bild der Galerie geladen wird. Falls er sich dazu entscheidet, die Galerie zu besuchen, obwohl noch nicht alle Bilder geladen sind – auch kein Problem: Wir setzen über jedes Bild eine kleine Loading-Animation, die erst verschwindet, wenn das darunter liegende Bild geladen und damit zu öffnen ist.

Zusätzlich haben wir uns für den Bildaufbau einen visuellen Effekt einfallen lassen (Abbildung 4), der nicht nur schön aussieht und einer der Highlights der Seite ist, sondern auch nur zwischen ca. 7 und 14 Sekunden (je nach CPU-Leistung, denn sie ist in der Sättigung) braucht, bis ein Bild wirklich vollständig gezeigt wird.

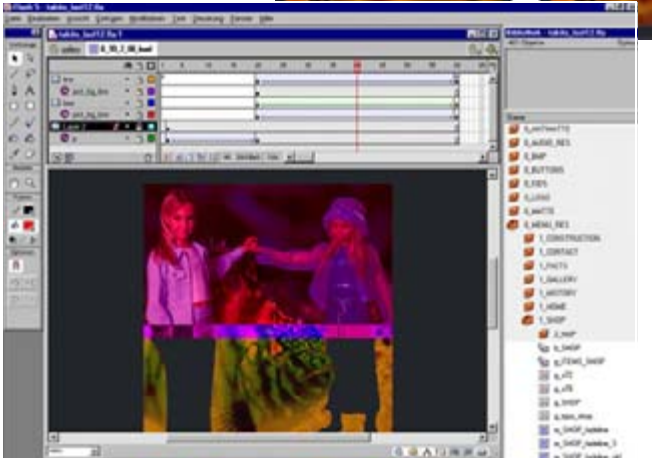
Wenn wir also vom schlimmsten Fall ausgehen (das ist in diesem Beispiel ein Besucher, der sofort nach dem Ladevorgang der Hauptseite auf die Galerie wechselt und darüber hinaus über eine sehr langsame Inter-

Abbildung 2 ▶
Taki-To-Homepage –
Preload der Hauptinhalte



◀ Abbildung 3
Taki-To-Homepage –
Preload der Galerie

Abbildung 4 ▶
Taki-To-Homepage –
Ablauf der Galerie der
Animationen



◀ Abbildung 5
Taki-To-Homepage –
Entwicklungsumgebung der
Galerie der Animationen

netanbindung verfügt), so überbrücken wir durch diesen Animationseffekt wieder einige Sekunden.

CPU

Wenn die CPU in der Sättigung ist (Auslastung bei 100 %), dann geht auch die Lade-Performance gegen null. Um den gewünschten Effekt zu erreichen, müssen Sie »Austastlücken« in Ihre Animation einbauen. Austastlücken sind kleine Abstände von wenigen Frames, in denen Ihre Flash-Animation keine CPU benötigt.

Der Animationseffekt basiert auf zwei Bildern. Das erste enthält den Hintergrund ohne die Kinder, das zweite nur die Kinder. Diese zwei Bilder werden nun über eine Maskenanimation (Streifen, der von oben nach unten läuft) übereinander geblendet und verschmelzen dadurch. Damit wir der CPU endgültig den Rest geben, haben wir zusätzlich noch auf »erweiterte Farbeffekte« zugegriffen, die wir über verschiedene Tweens zusätzlich über unsere Bilder laufen lassen.

Wie das innerhalb der Entwicklungsumgebung etwas vereinfacht (ohne Austastlücken) aussieht, sehen Sie in Abbildung 5.

www.die-waescherei.com

◀ Gerald Marischka

Schon Mitte 1999 ist diese Seite entstanden, als Flash 3 noch das Maß aller Dinge war. Heute sprechen nur noch Kompatibilitätsüberlegungen dagegen, ein Flash 5-Update zu machen. Ein Vorteil wäre natürlich, dass ein ein Update auf Flash 4 oder Flash 5 sicherlich zu einer beträchtlichen Ladezeitverkürzung führen würde (Flash 3 konnte Sounds nur im .wav-Format ausgeben, und die Texte konnten noch nicht aus einer .txt-Datei geladen werden).

Manchmal ist es nicht schlecht, auf »alte« Technologie zu setzen, um die Kompatibilität der Seite der Zielgruppe anzupassen. Mit einigen kleinen Tricks kann man dennoch eine herausragende Seite gestalten. Was dabei entstehen kann, möchte ich Ihnen anhand der Wäscherei-Seite beschreiben.

Simple is best. Die Wäscherei-Homepage zählt zwar heute sicher nicht mehr zu den technischen Highlights des Webs, ist allerdings auf Grund ihres Designs (von Gustav Assem) und der Kompatibilität nach wie vor ein Renner.

Abbildung 6 ▶
Die Wäscherei-Homepage

▲ Abbildung 7
Die Wäscherei – Interaktionen

Das User-Interface

Beim User-Interface haben wir uns dafür entschieden, möglichst oft MouseOver-Actions zu verwenden, um eine noch größere Interaktivität zu erzeugen. Diese Entscheidung muss mit Vorsicht genossen werden, da man die Seitenbesucher oft verwirrt.

In unserem Fall »spielen« wir mit dieser Verwirrung und damit auch mit den Besuchern. Zusätzlich erreichen wir so den Eindruck einer noch schnelleren Seite. Der Besucher ist also gezwungen, sich mit für ihn neuen Möglichkeiten der Interaktion zu befassen. Für unsere Zielgruppe erwies sich dies als optimal. So kommt es, dass wir nur bei den Hauptmenüpunkten und anderen ladeintensiven Vorgängen (Bilder) eine Release-Action verwendet haben und praktisch alle Untermenüpunkte aus MouseOvern bestehen.

Bei den verwendeten Formularen (Reservierung und Gästebuch) verwenden wir HTML-Pop-ups, die man seit Flash 4 als Flash-Formular in die

◀ **Abbildung 8**
Die Wäscherei – Reservierung mittels HTML-Pop-up

Seite einarbeiten könnte. Bei der Reservierung übergeben wir die Variable der Tischnummer mittels JavaScript an das Pop-up.
Jeder Tisch hat folgende `Button Release`-Action:

```
on (release) {
   getURL ("javascript:BestellTisch('tischnummer');");
}
```

Hierbei muss `tischnummer` durch die tatsächliche Nummer des Tisches ersetzt werden. Im Quellcode der übergeordneten HTML-Datei (Datei, die das SWF aufruft) muss nun noch das betreffende JavaScript eingesetzt werden.

```
<script language="JavaScript">
function BestellTisch(tischnr)
      {
      Inhalt = "order.shtml?" + tischnr;
      window.open(Inhalt,"TischReservierung","screenX=0,
screenY=200, resizable=1, location=0, directories=0, status=0, menu-
bar=0, scrollbars=0, toolbar=0, width=350, height=470");
      }
</script>
```

Abbildung 9 ▸
HomePage
Flash-Spiele

Die angegebene Datei order.shtml ist unser Pop-up-Fenster. Damit wir nun im Pop-up auch unsere Tischnummer ausgeben können, benötigen wir den Eintrag

```
<input type="Text" name="TischNr." value="<!--#echo
var="QUERY_STRING"-->
```

Nun fügen wir noch die Eingabefelder der zusätzlich benötigten Daten (z. B. Name, Uhrzeit usw.) ein, und schon ist unser Bestellformular fertig.

Um nun noch das Tüpfelchen auf das i zu setzen, hat Christian Pilsl das vorhandene Logo nicht nur gedreht animiert, sondern mehrere Masken und Alphavarianten übereinander gesetzt. Dadurch wirkt das Logo plastischer und gibt der sonst eher flach gehaltenen 2D-Seite einen Art 3D-Gegenpart.

www.flashspiele.de

▸
Peter Krempl
www.flashspiele.de

Auf der Website von Flashspiele.de werden verschiedene in Flash 4 realisierte Spiele gezeigt. Regelmäßig entstehen in der Keramikwerkstatt Joglhof im Bayerischen Wald (www.keramikwerkstatt.de) neben Brottöpfen

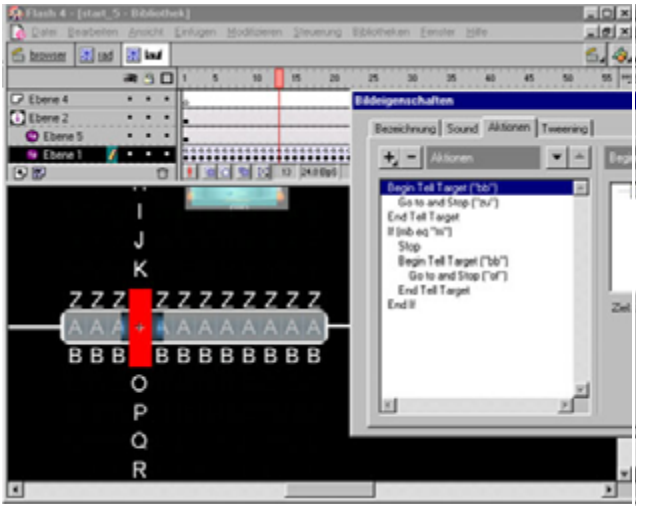
und Bierkrügen verschiedene Spiele in Flash. Wie es in Bayern halt üblich ist: Laptop und Latzhose.

Die auf der Seite gezeigten Spiele reichen vom Poolbillard über Karambolage, Memory, animierte Puzzels, Snake, Fußball und verschiedene »Ballerspiele« bis zu Kinderspielen wie einem Malspiel oder dem Komponierspiel, in dem (nicht nur) Kinder auf einem virtuellen Piano eine Melodie komponieren und abspielen können.

Eigentlich war erst keine Flash-Seite für die Spiele geplant. Es schien zu umständlich, die Flashseite bei jedem neuen Spiel zu erneuern. Eine reine HTML-Seite schien praktikabler. Da Flash-Spiele nun aber auch in einen Flash-Rahmen gehören, wurde für die Präsentation eine Mischform gewählt. Die Spiele öffnen einzeln in eigenen kleinen Pop-up-Fenstern, die aus Flash heraus gestartet werden. Dazu werden die verschiedenen Screenshots auf der Flashsite angeklickt. Die Textbezeichnungen für die Spiele werden in einem Textroller erzeugt. Die Button Action ruft nur das JavaScript für das Pop-up-Fenster auf und übermittelt den Namen des Spieles an den Textroller, der dazu neu gestartet wird und den Namen des Spieles anzeigt. Kommt ein neues Spiel in die Seite dazu, wird nur ein Bild eingefügt und ein unsichtbarer Button darüber gelegt.

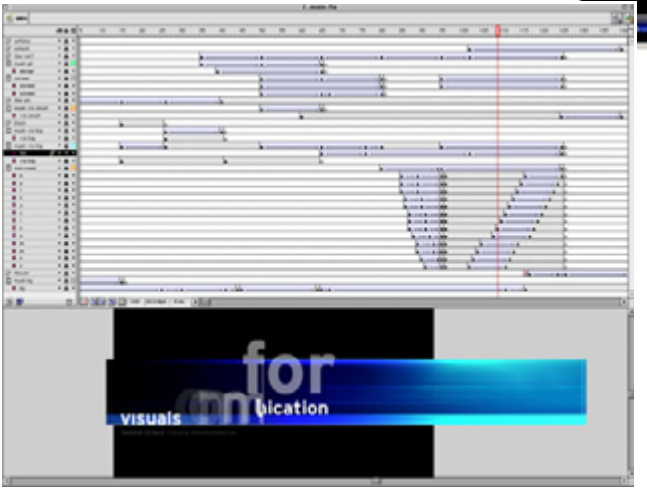
Mit der gewählten Methode können Aktualisierungen schnell bewerkstelligt werden. Die Spiele werden auf der HTML- und auf der Flash-Seite mit denselben JavaScript-Pop-up-Fenstern geöffnet. Sie liegen nur in einer Ausführung auf dem Server. Die Pop-up-Fenster erlauben es, die Darstellungsgröße der Spiele nachträglich zu verändern. Dies ist bei auftretenden Geschwindigkeitsproblemen von Bedeutung. Je kleiner ein Flashspiel auf dem Screen dargestellt wird, umso schneller kann es ablaufen. Alternativ hätte man sie auch mit LoadMovie in die Flash-Seite laden können, hierbei wären spätere Manipulationen an der Größe jedoch nicht mehr möglich gewesen.

Der Textroller selbst besteht aus zwölf Instanzen eines MovieClip. In diesem MovieClip laufen jeweils alle Buchstaben in einer Art Buchstaben-Slotmachine durch das ganze Alphabet, bis sie »ihren« Buchstaben erreicht haben. Diesen erhalten sie, indem mit Substring jeder Buchstabe aus einer von der `Button`-Action vorgegebenen Variablen ausgelesen wird. Da es unterschiedlich lange dauert, bis jeder Buchstabe erreicht ist, rasten die Buchstaben nacheinander ein; eben wie bei einer Slotmachine.



▲ **Abbildung 10**
Flashspiele.de – der Textroller

Abbildung 11 ▶
Das Online-Portfolio
www.schech.net



◀ **Abbildung 12**
Die Schech.net-Intro-Szene:
Die gestreckte Hintergrundgrafik wird entlang der x-Achse verschoben.

www.schech.net

◀
Dominik Schech
Visuelle Kommunikation

Diese Flashsite wurde Ende 1999 mit Flash 4 erstellt. Als Ergebnis erhielt ich nicht nur eine interaktive Website, sondern auch eine Hybrid-CD-ROM für Mac und PC. Mit kaum einem anderen Tool lässt sich so unproblematisch eine plattformunabhängige Präsentation umsetzen.

Im Bemühen um Abwärtskompatibilität verzichtete ich auf den Einsatz von ActionScript-Features wie Variablen und arbeitete ausschließlich mit Flash 3-kompatiblen Befehlen.

Die Site teilt sich in zwei Szenen: das Intro und der Inhalt.

Im Intro finden hauptsächlich Symbole und Bilddaten Verwendung, die später auch im Contentbereich benötigt werden. Aus diesem Grund tragen sie nur unwesentlich zur Erhöhung der Dateigröße bei. Vielmehr dienen sie zur Überbrückung von Ladezeiten. Ein erster Preloader zu Beginn ermöglicht einen schnellen Start, nachdem das halbe Intro geladen ist. Ein zweiter Preloader am Ende des Intros fängt mögliche Bandbreitenengpässe ab.

Das Intro besteht aus einer Vielzahl von Maskenanimationen und einfachen Tweenings. Um beim Erstellen die Animationen besser kontrollieren zu können, verzichtete ich weitgehend auf den Einsatz von Movie-Clips. Interessante Effekte ergeben sich durch horizontales Skalieren und Verschieben des Hintergrund-PNGs. Allerdings können solche Bitmap-Animationen schnell zum Performance-Killer ausarten, weshalb immer folgende Regel Beachtung finden sollte: Entweder man verwendet Alpha-Tweenings ohne Bewegung (wie zum Beispiel bei der Blume am Ende des Intros), oder man bewegt die Bitmaps und verzichtet auf Alpha-Tweenings. Allerdings sollten die Bitmaps auch hier auf einen Alphawert von 99 % gestellt werden – die Verwendung von Bitmaps mit einem Alphawert von 100 % führt zu Darstellungsfehlern.

Content

In der zweiten Szene befindet sich der wesentliche Content, bestehend aus den drei Teilbereichen Kontakt, Info und Portfolio. Diese drei Teilbereiche bestehen wiederum aus Startlabel, Fade-in-Animation, Inhalt und Fade-out-Animation. In Photoshop wurden die Hintergrundgrafik sowie die darüber liegenden kleineren Teile mit content-bezogenen Inhalten (Telefon, Blume, Foto) angelegt und als PNG in Flash importiert.

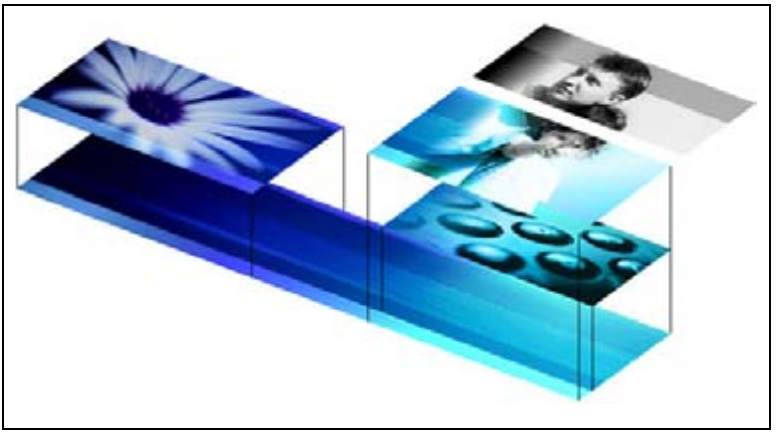


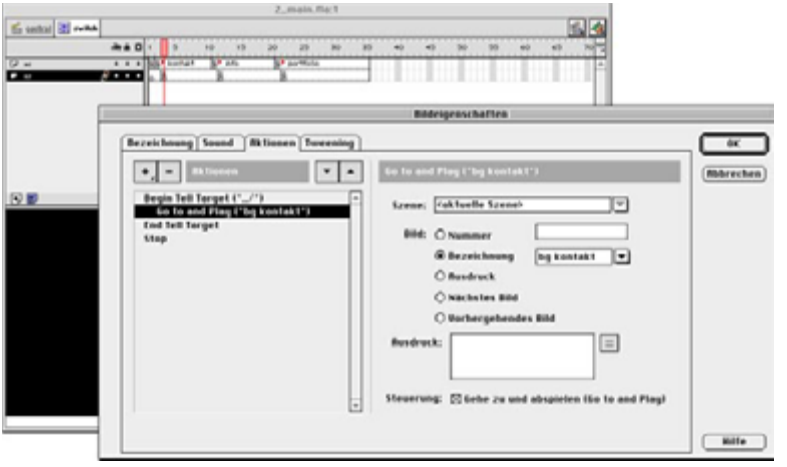
Abbildung 13 ▶
Schech.net – deckungsgleiche Grafiken wurden mit Photoshop vorbereitet.

Versuche, die Content-Grafiken mit Alphakanälen weicher freizustellen, führten zu größeren Dateien.

Drei Buttons, von denen innerhalb der Teilbereiche jeweils zwei aktiv sind, bilden die Navigation. Beim Anklicken eines Buttons wird das Abspielen auf der Haupt-Timeline fortgesetzt, die Verzweigung geschieht erst am Ende der Fade-out-Animation. Die Information, wohin verzweigt wird, muß also zwischengespeichert werden. Eine Lösung mit Variablen ist allerdings nicht abwärtskompatibel, eine Lösung mit dem Befehl Tell Target dagegen schon. Ein leerer MovieClip mit dem Instanzennamen `switch` dient als verzögerter Schalter.

Dieser MovieClip befindet sich auf einem Extralayer und besteht aus sieben Schlüsselbildern. Ein Stop in Frame 1 verhindert das Ausführen nachfolgender Befehle. Drei Label dienen zum Anspringen durch den jeweiligen Button, im Frame hinter dem Label befindet sich der Tell-Target-Sprungbefehl für die jeweiligen Teilbereiche auf der Hauptzeitleiste. Wird nun ein Button gedrückt, erhält der Clip »switch« einen Sprungbefehl, und ein Stopp zum entsprechenden Label, und die Haupt-Timeline erhält den Befehl, das Abspielen fortzusetzen:

```
On (Release)
   Begin Tell Target ("/switch")
      Go to and Stop ("portfolio")
   End Tell Target
   Play
End On
```



◀ **Abbildung 14**
Schech.net – Switch, ein leerer MovieClip dient als Schalter

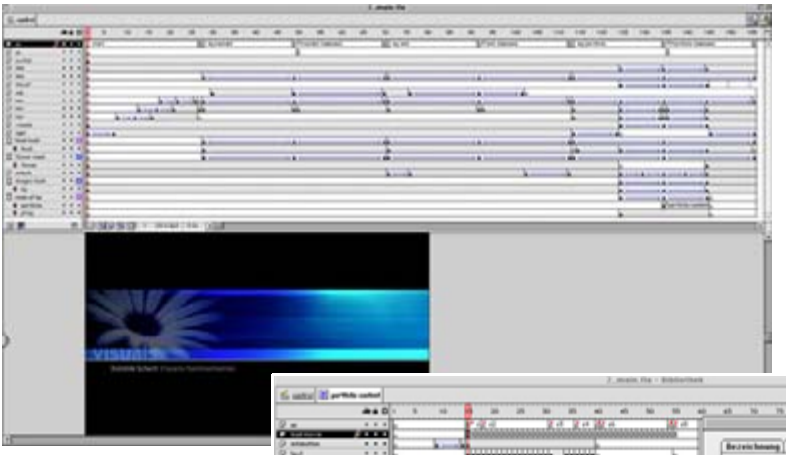
Am Ende der Fade-out-Animation führt folgende Befehlsfolge zum Verzweigen:

```
Begin Tell Target ("/switch")
   Play
End Tell Target
Stop
```

Der MovieClip `switch` kommt zum eigentlichen Verzweigungsbefehl, das Abspielen wird beim Fade-in des Teilbereiches `Portfolio` fortgesetzt:

```
Begin Tell Target ("../")
   Go to and Play ("bg portfolio")
End Tell Target
Stop
```

Die Inhalte »Kontakt« und »Info« liegen direkt auf der Hauptzeitleiste, das komplexere Portfolio in einem eigenen MovieClip. Komplexe Inhalte sind in separaten MovieClips besser aufgehoben, Änderungen und Tweening-Korrekturen gestalten sich wesentlich einfacher.



◂ **Abbildung 15**
Schech.net –
Die Content-Szene

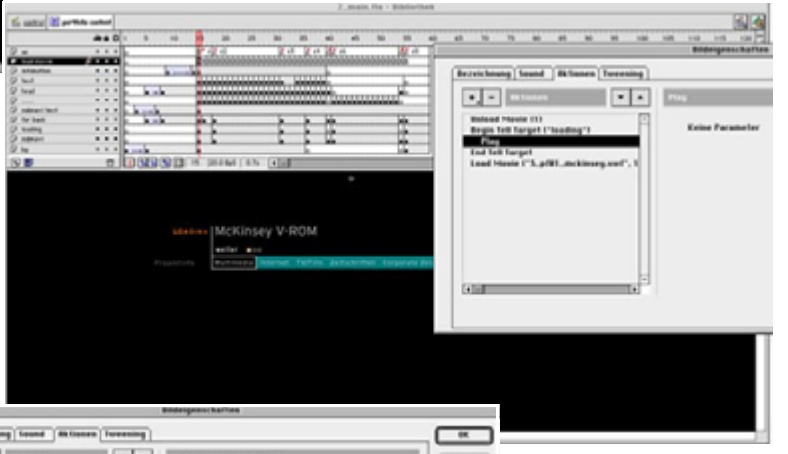


Abbildung 16 ▸
Schech.net – Der
Portfolio-MovieClip

◂ **Abbildung 17**
Schech.net – Ein spezieller Flash-Film enthält die Bilddaten und meldet dem Hauptfilm seinen Ladezustand.

Portfolio-MovieClip

Der Portfolio-MovieClip besteht aus dem Einblenden des Layouts sowie der eigentlichen Funktionalität. Jede Bildposition auf der Zeitleiste entspricht einem Inhalt im Portfolio. Sich verändernde und gleich bleibende Inhalte liegen getrennt auf entsprechenden Ebenen. Alle Bildinhalte liegen in Extradateien, die mittels Load Movie auf Level 2 geladen werden. Sämtliche Texte und Headlines sind allerdings im Portfolio-MovieClip enthalten – dies führt in der Summe zu geringerer Dateigröße. Flash wandelt eine verwendete Schrift beim Export intern in Symbole um. Zur Datenreduktion wird dabei jeder Buchstabe nur einmal als Grafik gespeichert und mit Instanzen dieser Grafik gearbeitet. Wäre der Text jeweils im zu ladenden Film, würde mit ihm auch die Schrift jeweils neu geladen.

Mittels der Buttons Zurück und Weiter kann man sich bildweise durch das Portfolio bewegen. Jede Rubrik kann aber auch direkt durch den Sprung auf ein Label angesteuert werden.

Ein MovieClip mit Loading-Hinweis wird vom geladenen Bild-SWF mittels Tell Target auf ein leeres Schlüsselbild gesetzt, sobald der Ladevorgang beendet ist. Dazu wird im Bild-SWF kein eigener Preloader benötigt. Es reicht, den Tell-Target-Befehl in einen Frame hinter der Bitmap-Datei zu vergeben, da der Film diesen erst nach dem Laden der Bilddateien erreicht und damit den Befehl auslöst.

Diese Art des Zusammenspiels zwischen geladenem Film und Hauptfilm könnte natürlich noch weiter verfeinert werden. Ein Beispiel dafür ist die Seite www.stefanboehme.net, das Portfolio eines Werbefotografen. Die Bilddaten sind wesentlich größer als auf www.schech.net. Hier liegen alle Bilder einer Rubrik in einem SWF, Flash streamt die Datei, d. h. alle Bilder werden hintereinander geladen. Eine verschachtelte `IfFrameisLoad`-Abfrage gibt auf der Haupt-Timeline pro Bild einen Button frei. So können die bereits geladenen Fotos betrachtet werden, während Flash im Hintergrund fleißig weiterlädt.

www.toferer.at

◀ Belinda Baumgartner – Algo Software

Online-Shopping, Internetpräsenz, Imagepflege – Schlagwörter, die in letzter Zeit immer mehr an Bedeutung gewonnen haben. In großen Ballungs- und Wirtschaftszentren schon Standard, so wurde der erfolgreiche, sorgfältig geplante Internetauftritt von Unternehmen in Randregionen bisher noch etwas vernachlässigt.

Abbildung 18 ▶
Startseite –
Auswahlbereich

»Flash – Was ist das ?« Die Möglichkeit, sein Unternehmen nicht nur statisch im Internet zu präsentieren, sondern dem Kunden die Angebotspalette dynamisch näher zu bringen, kannten die wenigsten.

In den letzten Monaten konnten wir jedoch eine verstärkte Nachfrage insbesondere nach Flash-Auftritten verzeichnen. Überraschend viele Nachfragen kamen dabei besonders von kleinen und mittelgroßen Unternehmen, wo man bisher das Internet nicht bzw. in geringem Ausmaß nutzte.

Der Kunde, für den wir diese Flashsite erstellt haben, ist ein Groß- und Einzelhändler mit eigener Kollektion im Bekleidungsbereich, der erstmals sein Unternehmen im Internet präsentieren wollte.

Der Kunde wollte eine Flashsite (Flash 4) mit integriertem Shopsystem auf PHP3-Basis. Das Shopsystem (zum Zeitpunkt der Veröffentlichung dieses Buches noch im Aufbau) wurde von einer Partnerfirma realisiert.

Wir bekamen vom Kunden als Vorgabe einen Firmenprospekt und die Aufgabenstellung, eine möglichst einfach zu bedienende und dynamische, jedoch nicht zu aufwändige Website unter Berücksichtigung des CI zu erstellen. Hauptaugenmerk sollte dabei auf die Produktfotos gelegt werden, sie sollten möglichst gut die Angebotspalette des Unternehmens veranschaulichen.

Die Website mit Shopsystem sollte vor allem die Wiederverkäufer – ein Großteil der Kunden der Fa. Toferer – ansprechen. Die Bestellfunktion sollte leicht verständlich sein und auch grafisch ansprechend gestaltet werden.

◂ **Abbildung 19**
Jede Warenart erhält ihr eigenes Movie.

◂ **Abbildung 20**
Bereich T-Shirts – Auswahl mit scrollbarer Bildleiste. Die Darstellung der Produktfotos ist ein Kompromiss zwischen Bildqualität und Ladezeit.

Als Richtwert für die Ladezeiten nahmen wir eine Übertragungsrate von minimal 56 kBit/s an.

Eines der Hauptprobleme, das wir meistern mussten, waren die vielen Produktfotos: Sie sollten einerseits nicht zu große Ladezeiten verursachen, andererseits durfte aber auch die Bildqualität nicht zu sehr unter der Kompressionsrate leiden.

Die Filmgröße sollte laut Wunsch des Kunden unbedingt für jede Browsergröße skalierbar sein, was natürlich auch bei der Bildbearbeitung berücksichtigt werden musste.

Intro/Opener

Auf ein langes Intro verzichteten wir auf Grund der nicht gerade kurzen Ladezeiten in den Produktbereichen. Stattdessen erstellten wir einen kurzen Opener, in dem wir passend zum Gesamtbild der Site und des CI zuerst das Firmenlogo und den Slogan sowie die Navigationselemente aufbauend animierten.

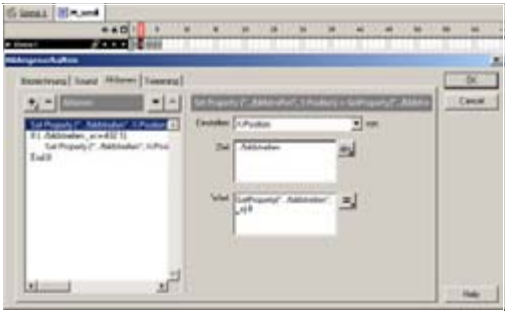
Dabei verwendeten wir hauptsächlich Alpha-Tweenings sowie effektvolle Maskenanimationen (Linien). Wir erstellten dann für jede Warenart ein eigenes Movie, das per LoadMovie in verschiedene Ebenen geladen wird.

Scroll-Bildleiste

Da es relativ viele Fotos pro Warenart gab, entschieden wir uns für eine horizontal scrollbare Bildleiste.

Abbildung 21 ▶
Zoom-Funktion in der mittleren Web-Qualität

Abbildung 22 ▶
ActionScript-Fenster mit Scrollfunktion für die Bildleiste



Die Produktfotos erstellten wir optimiert auf eine maximale Browsergröße von 1.024 x 768 im Bitmap-Dateiformat. Mit diesem Format stellten wir sicher, dass Flash die Fotos in der optimalen Qualität exportiert, wobei eine doppelte Kompression wie z. B. bei der Verwendung von GIFs oder JPEGs vermieden wird.

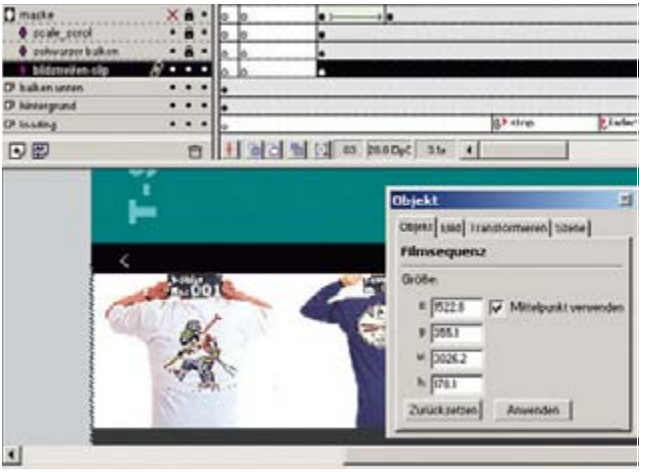
Nehmen wir z. B. die Warenart »Sweats«:

Zuerst erstellten wir für jedes Produktfoto einen MovieClip (nicht Grafik, um den so genannten Grafikbug zu umgehen), danach setzten wir jeden dieser MovieClips in einen Button und diesen Button wiederum in einen MovieClip.

Wir hatten jetzt für jedes Sweater-Foto einen MovieClip, in dem ein Button steckte. In diesen MovieClip setzten wir zusätzlich einen MovieClip namens `Scaler` – dieser sollte die Skalierung der Fotos beim Buttonklick ermöglichen. Diese Skalierung erfolgt per ActionScript mit `SetProperty/X-Skalierung` sowie `Y-Skalierung`.

Wir setzten nun diese MovieClips mit den Fotos nebeneinander auf einen entsprechenden Hintergrund und ordneten sie so an, dass links und rechts noch genügend Platz für die Skalierung war. Diese Bildleiste legten wir in einem MovieClip an, den wir auf die Haupt-Timeline des Sweater-SWFs legten.

Nun benötigten wir noch die Scrollpfeile links/rechts sowie einen MovieClip `M_scroll`, der die Bildleiste per ActionScript bewegt und der das Script hierfür beinhaltet.



◀ **Abbildung 23**
Festlegung der Scroll-Koordinaten

Diesen Scroller legten wir ebenso auf die Haupt-Timeline unseres SWFs. Nun mussten wir nur noch die Maximalkoordinaten der Bildleiste ermitteln. Dafür bewegten wir die Bildleiste von der Haupt-Timeline aus an ihr linkes bzw. rechtes Ende und notierten die entsprechenden X-Koordinaten.

Diese bauten wir dann in das Script der Scrollbuttons ein, und zwar so, dass sich die Bildleiste immer so weit bewegte, bis sie ihren Endpunkt auf der jeweiligen Seite erreicht hatte. Voilà!

Diese Site ist sicher interessant für den Flasher, der lieber händisch animiert als ActionScript programmiert. Ich persönlich zähle mich auch eher zu den Ersteren, obwohl gerade Flash 5 für ActionScript-Liebhaber ein Paradies ist und mich ebenfalls langsam aber sicher mit seinen schier unendlichen Möglichkeiten im ActionScript-Bereich in seinen Bann zieht.

Elsa – das Intro

www.powerflasher.de/elsa/intro1

◀
Carlo Blatz

Anfang 2000 kam der Kunde Elsa auf uns zu, ein innovatives Intro für den Internetauftritt der Firma zu produzieren. Vorgabe war, etwas Effektvolles zu gestalten, etwas unreal und sehr verspielt – wie der Printauftritt. Hintergrund ist die Zielgruppe »Gamer« – Elsa stellt unter anderem Grafikkarten für hohe Spielansprüche her.

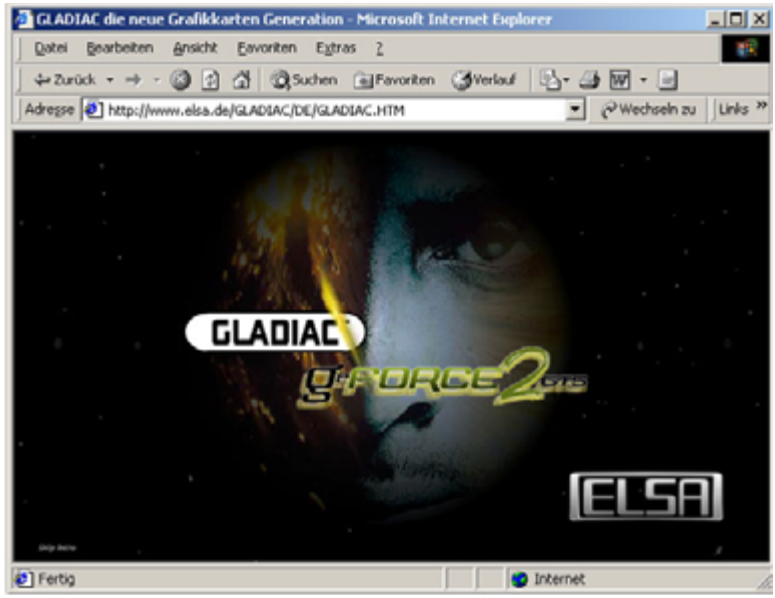
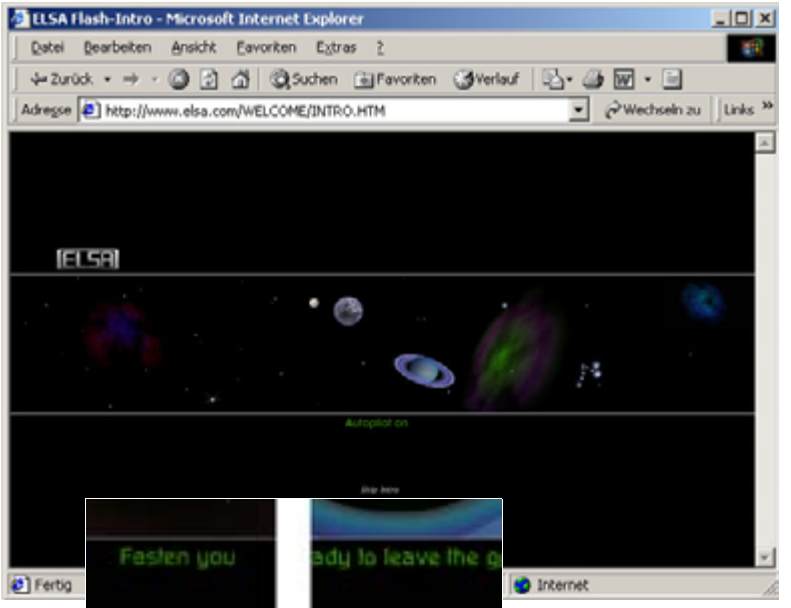


Abbildung 24 ▶
Gladiac Produkt-Push – Fullscreen

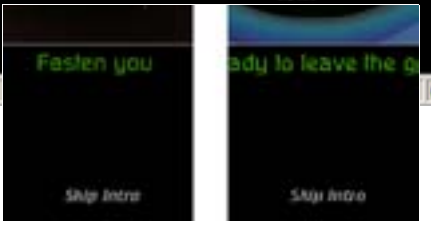
Die Zielgruppe war also bereits vorgegeben, eine Serverauswertung dokumentierte einen seltenen, unglaublichen Anteil von 90 % Flash 4-Usern – wir konnten also mit dem Konzept beginnen. Über den Ablauf bis zur Fertigstellung habe ich bereits in den anderen Kapiteln geschrieben: Briefings, diverse Freigabesteps etc. (siehe Kapitel »Projektmanagement« auf S. 20), doch welche Schwierigkeiten gab es, bis wir zu dem Ergebnis kamen, wie es heute zu sehen ist!

Der Film wurde bewusst aufwändig gestrickt – Ziel war es, möglichst viel Aufmerksamkeit zu erzeugen. Sowohl in der Dateigröße als auch in der Performance schlägt sich dieses Ziel nieder. Um das Problem der Performance in den Griff zu bekommen, haben wir den Bildschirm begrenzt. Im Gegensatz zu www.powerflasher.de/elsa/gladiac (eine Produktpräsentation) ist das Intro nicht Fullscreen.

Ein Pop-up in bestimmter Größe kam nicht in Frage, da dies dem Introeindruck nicht entspricht. Das gewählte Breitbandformat hat auch einen weiteren Vorteil. Die Sicht des Menschen entspricht nicht dem 4:3-Format, wie man es vom Browser gewohnt ist. Der Mensch sieht deutlich breiter als höher (daher auch 16:9). Das Intro ist so also schneller und wirkt optisch angenehmer.



◂ **Abbildung 25**
Elsa – das Intro mit vertikaler Begrenzung



◂ **Abbildung 26**
Die Qualität wird heimlich umgeschaltet

Ein weiterer Trick, um die Performance zu steigern, wird vor dem Tunnelflug angewendet. Der Tunnelflug ist eine ActionScript-Animation, die durch ein `DuplicateMovieClip` erzeugt wird. Erste Versuche sahen noch spektakulärer aus, da die Ringe eine Alphatransparenz an den Rändern hatten. Leider hat das kein Rechner mehr flüssig geschafft. Da wir nun schon Vektoren mit harten Kanten hatten, konnten wir das Antialiasing minimieren, ohne dass es einen sichtbaren Unterschied gibt. Lediglich am SkipIntro-Button enttarnt sich der Trick. Erst kurz vor dem Ticker »Autopilot on« wird wieder auf High Quality umgeschaltet und das Antialiasing wieder aktiviert.

Das Problem der Dateigröße war hier nicht allzu schwerwiegend. 57.600 bps wurde als Minimum angesetzt. Wir haben dennoch zur Sicherheit das Streaming auf 28.800 bps optimiert. Ideal hat ein Intro keine Ladezeiten, sondern versüßt die Ladezeit der eigentlichen Homepage. Bei den Ansprüchen an dieses Intro war das nicht möglich. Nonstop Sound, 3D-Animationen und Pixelbilder machen bei mehr als einer Minute über 600 KB aus und würden ein flüssiges Streaming erst bei DSL zulassen. Wir

mussten also einen Preloader vorschalten, der allerdings nicht 100 % der Site vorladen sollte. Zuerst wird die Übertragungsgeschwindigkeit gemessen, indem die Zeit gestoppt wird, die es dauert, die ersten 10 KB zu übertragen. Damit können wir auf die Übertragungsrate schließen und wissen, wie viel Prozent der Site vorgeladen werden müssen, um das Intro ohne Nachladezeit flüssig darzustellen. Mehr dazu unter »Preloader« auf Seite 610.

Ein Nachladen, während das Intro bereits läuft, würde ein unangenehmes Stocken verursachen. Im schlechtesten Fall (28.800 bps – 2,4 KB/s) müssen immerhin 60 % des Files vorgeladen werden. Das macht 360 KB bei gut zwei Minuten Ladezeit. Mit ISDN sind es allerdings nur wenige Sekunden.

Der Sound macht fast 200 KB des Intros aus. Damit dieser nicht vorgeladen werden muss, steht die Synchronisierung auf Streaming. Natürlich vorrangig auch, damit die Animationen in jedem Frame soundsynchron sind. Streamingsounds haben allerdings einen unerwünschten Nebeneffekt. Animationen können nicht langsamer werden, sondern nur ruckeliger. Flash spielt den Sound ab und stellt dabei nur die Bilder dar, die der Prozessor noch schafft. Bei einer Bildrate von 20 Bildern pro Sekunde hat Flash also 20 Mal in der Sekunde die Chance, ein Bild darzustellen. Um Flash mehr Chancen zu geben, haben wir die Framerate auf 40 Bilder pro Sekunde hoch gesetzt. Zwar sind maximal 25 Bilder pro Sekunde für das Auge flüssig, aber um diese überhaupt zu erreichen, haben wir die Framerate hoch gesetzt.

Der Rest erklärt sich fast beim Zusehen. Vielleicht ein paar Daten:

4000 Frames, 340 Symbole, eine Sounddatei, eine Szene, vier Wochen Entwicklungszeit, zwei Flasher, ein Grafiker, ein 3D-Artist.

Etwas knifflig ist noch der Preloader-Texteffekt. Es handelt sich um ein Textfeld, in dem die Schriftkonturen eingebunden sind, damit wir den Text überhaupt animieren können. Das Textfeld liegt in einem MovieClip, dessen Instanz wir ansprechen und animieren können. Das System dieses Effekts wird ausführlich unter »Effekte mit Duplicate Movie« auf Seite 582 erklärt.

◀ **Abbildung 27**
Videotrick

◀ **Abbildung 28**
MovieClip um zwei Frames versetzt – 50 % Alpha

SSX – Electronic Arts

www.powerflasher.de/ea/ssx

◀
Carlo Blatz

Alle, die hoffen, dass wir diesen Sourcecode jetzt auch noch veröffentlichen, muss ich leider enttäuschen. Ich möchte an diesem Beispiel nur einen Trick zeigen, den man im Outro findet.

Nach Spielende sieht der User einen kurzen Trailer. Hier haben wir orginale Videosequenzen aus dem Playstation 2-Spiel eingebaut. Die Bilder wurden als BMPs ausgespielt und in Flash als JPGs komprimiert. Insgesamt handelt es sich um etwa 100 Einzelbilder à ca. 5 KB. Bei einer Framerate von 20 bps haben wir ,damit nur fünf Sekunden. Um diese Animation etwas zu verlängern, haben wir nur jedes vierte Bild ausgespielt und zeigen in Flash auch nur nach drei Bildern ein neues Bild, um die Geschwindigkeit beizubehalten. So haben wir 20 Sekunden Animation lediglich mit einer ruckeligen Framerate von nur 5 bps. Also bedienen wir uns eines Tricks. Alle vier Bilder kommt ein neues Bild. Jedes zweite Bild setzen wir jetzt einfach das kommende Bild bereits mit 50 % Transparenz über das »alte« Bild. So ist jedes zweite Bild ein neues Bild. Vollbild, Halbbild, Vollbild etc.

Zudem haben wir, wie man in diesem Bild sieht, einen schwarzen Verlauf zu den Rändern über die Bilder gelegt, den wir in vier Schritten am Anfang und am Ende jeder Animation sanft ein- bzw. ausblenden. Um den Prozessor zu schonen, handelt es sich aber tatsächlich um vier Einzelbilder – ein Tweening wäre zu intensiv.

Abbildung 29 ▸
Links überlappt 50 % Alpha des Vorbilds – rechts nicht

Damit man nicht jedes zweite Bild von Hand transparent setzen muss, haben wir die ganze Videosequenz in eine Filmsequenz gelegt und setzen diese um zwei Frames versetzt eine Ebene höher auf 50 % Transparenz.

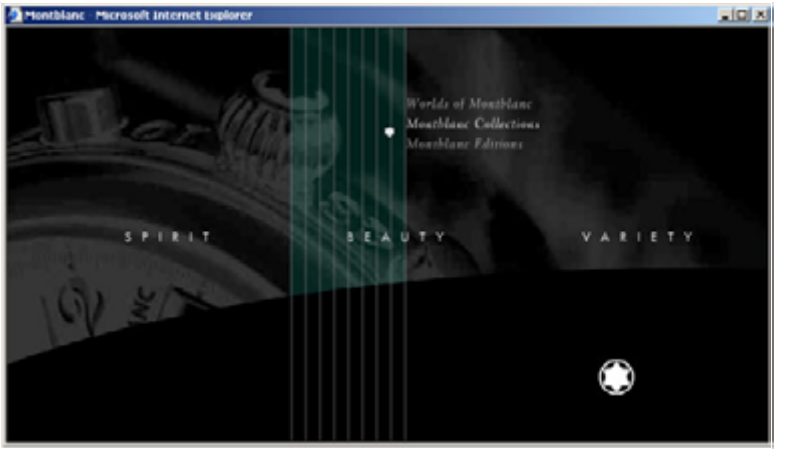
Auch im Elsa-Intro haben wir diesen Trick am Ende bei der Weltkugel angewendet. Sie besteht aus 30 Pixelbildern à 6 KB. Bei der Framerate von 40 bps reicht es, jedes dritte Bild ein neues Pixelbild zu zeigen, wobei alle Bilder zusammen eine Erdumrundung ausmachen. Dennoch sind 90 Frames nur etwas mehr als zwei Sekunden Animation für eine Drehung. Das war zu schnell, aber mehr Pixelbilder würden zu groß, und die 30 Bilder langsamer zu zeigen, würde stark ruckeln. Daher zeigen wir alle sechs Frames ein neues Bild, dazwischen aber auch wieder das nächste Bild mit 50 % Transparenz. Das macht einen weichen Effekt, dauert doppelt so lange und ist nur 10 KB größer.

www.montblanc.com

▸
Carlo Blatz

Mitte 2000 hat Elephant Seven (Springer & Jakoby) Powerflasher beauftragt, beim Relaunch der Internetpräsenz von Montblanc mitzuwirken. Wir haben die Realisation von zwei Produktsparten übernommen: Editions und Collections.

Der Kunde ist ein Traditionsunternehmen, das sich im höheren Preissegment mit edler Qualität platziert. Die Zielgruppe ist gediegen, und schnelle Effekte passen nicht. Wer denkt, dass das die Sache vereinfacht, hat sich getäuscht. Gerade dieser Umstand macht die Ansprüche und deren Umsetzung so kompliziert. Die Seite muss die Produktpalette würdig repräsentieren. Das gilt sowohl für Layout und Gestaltung als auch für die Animationen.



◀ Abbildung 30
www.montblanc.com

Ich möchte die Schwierigkeiten, den Weg und schließlich die Lösung etwas ausführlicher beschreiben. Es sind keine Innovationen, es beschreibt aber sehr deutlich, auf welch einfachen Wegen man manchmal zum Ziel kommt. Außerdem werden an diesem Beispiel viele konzeptionelle Punkte verdeutlicht, die wir in diesem Buch beschrieben haben.

Nach dem bewährten Motto »Less is more« haben wir die ersten Animationsdummys erstellt. Langsame, kurze, schlichte Animationen auf kurzen Wegen. Böse Zungen könnten jetzt einwerfen: »Warum dann überhaupt Flash?« Wenn wir die Produkte, Headlines etc. gar nicht einblenden würden, wäre das auch ein Bruch. Wenn auf Knopfdruck alle Bilder da wären, verlören wir den weichen, edlen Eindruck. Außerdem liegen uns relativ große Bilddaten vor, um die Produkte auch in angebrachter Qualität zu präsentieren. Diese Bilddaten wollen geladen werden, und da können wir uns bei Flash einesTricks bedienen.

Die Bühne

Aber fangen wir vorn an. Die Bühne ist ein Pop-up in der festen Größe 720 x 420, ein Wert, der auch bei einer Auflösung von 800 x 600 Pixel gut passt. Der Vorteil dieser Begrenzung ist, dass alle Bilddaten in Originalgröße importiert werden können und ohne Skalierung dargestellt werden. So gewähren wir beste Bildqualität und gehen keine Risiken ein, wie es bei verschiedenen Auflösungen in verschiedenen Browsern auf verschiedenen Systemen aussieht.

Abbildung 31 ▶
Montblanc Editions

Montblanc - Microsoft Internet Explorer
PATRON OF ART
WRITERS EDITION
SPECIAL EDITION
Hemingway
Agatha Christie
Imperial Dragon
Oscar Wilde
Voltaire
A. Dumas
F. M. Dostoevsky
Edgar Allen Poe
Marcel Proust
A MAN CAN BE BEATEN BUT NOT DESTROYED
HEMINGWAY
His philosophy pervaded his work, which culminated in the world-famous novel 'The Old Man and the Sea'. His terse and drastic language influenced the style of American literature. He became one of the greatest writers of our time: Ernest Hemingway.

Ladeverhalten

Die Daten haben wir in drei Teile aufgeteilt: die Navigation, die Textdaten und die Bilddaten. Diese drei SWFs werden von einer Startdatei, die nur dafür vorhanden ist, auf verschiedene Levels geladen.

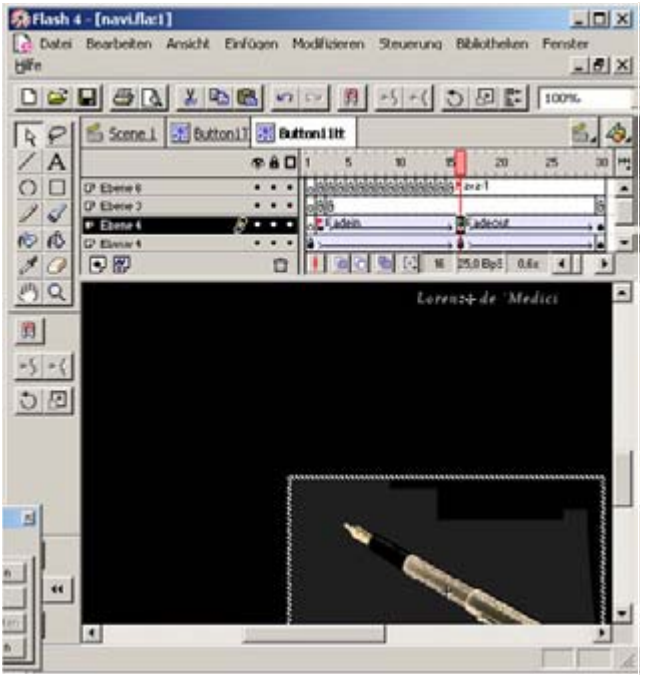
- Hintergrundbild auf Level 1
- Navigation auf Level 2
- Typo auf Level 3

Die Navigation und die Typo werden mit einem Preloader vorgeladen. Wenn diese Quellen vorhanden sind (etwa 20 KB), kann man die Seite ansehen. Das Hintergrundbild kommt etwas später, da man es ohne Preloader streamt. Es ist lediglich ein schwarzes Tweening sichtbar. Die Animation kann erst starten, wenn sie übertragen ist. Wir haben also durch das Streaming eine natürliche Verzögerung.

Bis hier hin sind beide Punkte – Editions und Collections – noch gleich.

Editions

Wenn der User, nachdem die Startseite geladen wurde, auf einen der drei Oberpunkte klickt, wird das Hintergrundstartbild gestartet – hier wird die Feder ausgeblendet, damit sie für den folgenden Text nicht im Weg steht. Die Typo wird angewiesen, zur entsprechenden Stelle zu springen, und die Navigation wird eingeblendet. Auch hier passiert nichts von einem Frame zum anderen, sondern stets mit Tweenings: Punkt für Punkt mit einer kleinen Überlappung.

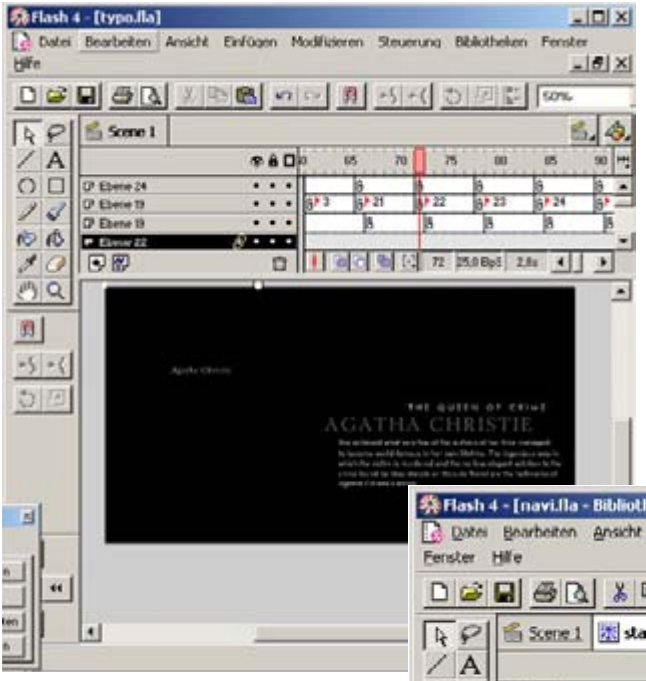


◀ **Abbildung 32**
Der etwas andere Button

Die Navigation

Hier wird's kompliziert. Damit auch die Navigation weich und edel anmutet, müssen wir das Rollover ebenfalls tweenen. Jetzt sollte man denken: »kein Problem – einfach eine Animation in den OverZustand des Buttons erstellen.« Was aber, wenn das Rollover markiert ist und der User den Button loslässt? Der Zustand wechselt wieder auf normal, und ohne Animation tauscht sich die Grafik aus – nicht sehr weich und edel. Zudem soll bei Rollover auch eine Vorschau des Produkts links unten erscheinen und natürlich weich sichtbar und wieder unsichtbar werden. Wir müssen also einen besseren Button bauen.

Jeder Button hat keine Zustände außer dem Aktivzustand. Jeder Button ist in einer Filmsequenz, die im ersten Frame gestoppt ist. Mit Rollover bekommt er die Aktion `Play`, wobei ein Tweening den Button in 15 Keyframes markiert. Im Endbild der Animation ist ein `Stop`. Soweit könnte man ja noch den Over-Zustand verwenden. Wenn der User nun aber ein Rollout macht, wird die Animation vom Highlighting wieder zum Ursprungszustand durch Tweening `Goto and play (15)` zurückgeholt.



◄ Abbildung 33
Das externe Typo-File

Abbildung 34 ►
Ein MovieClip nur für die Standings und Verdecker

Montblanc - Microsoft Internet Explorer
A SPORTING DESIGN
SPORT
COLLECTION
A Montblanc watch
makes a reliable partner for leisure time.
These chronometers do not only look sportive -
they keep their promises.

◄ Abbildung 35
Collections

Einziges Problem: Was, wenn der User ein Rollout macht, bevor die Animation vollständig markiert ist? Der Button würde, egal wie »hell« er schon ist, zum Endbild springen und dort wieder dunkel werden. Wir möchten aber, dass er exakt von diesem Stadium der Helligkeit wieder zurücktweent. Daher haben wir eine Laufvariable eingebaut, die in jeder Button-Filmsequenz die Keyframes zählt. Wenn die Rollover-Animation also startet, wird die Variable mit jedem Keyframe um 1 verändert. Somit wissen wir, wie weit sie bereits getweent ist, und können sie beim Rollout zu einem äquivalenten Punkt der Rollout-Animation schicken.

Die Theorie dahinter: Die Rollin-Animation besteht aus 15 Keyframes, die Rollout-Animation ebenfalls – insgesamt 30 Frames. Wir setzen die Zählervariable z also auf 30 und ziehen bei jedem Frame im Fade-in eins ab (z = z – 1). Folglich muss ein Tweening von zehn Frames im Rollin ab Frame 20 wieder ausrollen (30 – 10). Die Aktion lautet also: `On Rollout Goto and play (z)`.

Das funktioniert soweit auch sehr gut. Schwieriger wird es nun, weil auch noch die Aktion `on Release` hinzukommt. Wenn der User also auf den Button klickt, soll dieser natürlich aktiv bleiben und keine Rollover-Animation mehr starten. Zunächst versuchten wir, eine Variable zu setzen, die in Frame 15 abgefragt wird. Wenn sie gesetzt ist, soll der Button nach der Animation zwangsläufig stoppen. Der Vorteil dieser Version wäre, dass der Button soft zum gedrückten Zustand rollt und dort stehen bleibt. Das war aber etwas zu viel der Perfektion, je mehr man mit solchen Variablen spielt, desto anfälliger wird die Navigation auch. Flash verschluckt eventuell eine Variable, und schon sind zwei Buttons unabänderlich aktiv. Wir haben einen Trick verwendet und einfach in die Typodatei .swf, die eine Ebene höher geladen wird, den animierten Button als Grafik über die Navigation gelegt. Die eigentliche Navigation wird also nicht beeinflusst, sondern nur an dieser Stelle verdeckt (Abbildung 33).

Einen kleinen Bug gab es noch: Wenn man versehentlich beim Klicken die Maus bewegt, wird die Aktion nicht ausgeführt: Es handelt sich dann um kein `Release`, sondern um ein `ReleaseOutside`, was für die Button-Aktion ja etwas anderes ist. Die Aktion muss also `on Release` und `on ReleaseOutside` ausgeführt werden – ein Tipp, den man in fast allen Projekten beherzigen kann.

Aber nicht genug der Probleme: Die kleinen Vorschaubilder unten links befinden sich auch in der Button-Animation. Sie animieren sich also parallel mit dem Aktivieren der Buttons. Nachdem die Navigation aber schon perfekt funktionierte, wünschte sich der Kunde, dass die kleinen Rollover-Bildchen, wenn ein Button gedrückt wurde, auch stehen blei-

ben. Nun könnte man sagen, wir stellen sie doch einfach an die gleiche Position in der Typo. So einfach geht's aber leider nicht, denn dann wäre das kleine Bild, sobald der Button gedrückt wird, immer über der Navigation zu sehen, bis ein anderer Button gedrückt wird. Damit sieht man die Rollover-Bilder der anderen Buttons aber nicht mehr. Das Bild soll also nur zu sehen sein, wenn man über dem entsprechenden Button ist und über keinem anderen.

Wir haben also eine weitere Filmsequenz gebastelt, die unter der Navigation liegt und die bei jedem Buttonklick per Tell Target die Aktion annimmt: `GotoAndStop` zu einer Stelle, wo dieses Bild zu sehen ist. Da bei Rollover auf die anderen Buttons dieser nun auch wieder verschwinden muss – sonst überlagern sich zwei Federhalter –, haben wir hinter die transparenten Fotos eine graue Fläche gelegt: den so genannten Verdecker. Diese Fläche muss extrem genau zugeschnitten sein, weil sie keinen Teil der Navigation verdecken, aber auch keine Stelle des unten liegenden Bildes durchscheinen lassen darf. Bei niedrigen Farbwerten der Grafikkarte wie 16 Bit, enttarnt sich diese Fläche ganz leicht. Es entsteht während der Animation ein Mischwert aus dem Grau des Hintergrunds und dem Grau des Verdeckers, der von Alpha 0 auf 100 eingeblendet wird (Abbildung 34).

Jetzt zu Flash 5-Zeiten lächeln wir etwas über diesen mühsamen Weg, alle Kundenwünsche unter einen Hut zu bekommen. Mit Objects und Arrays gäbe es da viel einfachere Lösungen, damals war es aber ein durchaus praktikabler Weg.

Der Rest ist recht einfach und logisch. Beim Buttonklick wird die Typo per Tell Target angewiesen, zu einem entsprechenden Punkt zu springen, und dort wird ein LoadMovie des Hintergrundbilds eingeleitet. Dieses Hintergrundbild wird in Ruhe geladen, und wenn es da ist, erscheint es sanft aus dem Schwarz. Die durch die Ladezeit bedingte Verzögerung wirkt gewollt. Auf diese Weise müssen nicht alle Daten vorgeladen werden. Bei den großen Hintergrundbildern würde das ein paar Minuten Ladezeit ausmachen. So werden die Daten nur nach Bedarf geladen.

Collections

Vom Prinzip her ist die Technik in der Montblanc Collection die gleiche. Etwas einfacher wurde es durch den Vorschaustreifen. Er hat immer die gleiche Größe – Umwege wie den Verdecker bei Editions benötigen wir hier also nicht. Die Typo und die Hintergrundbilder funktionieren nach dem gleichen Prinzip. Neu ist der scrollbare Produktkatalog. Wenn ein Menüpunkt ausgewählt wurde, kann man an der rechten Seite alle Produkte dieser Serie durchscrollen. Sie werden natürlich nicht einfach nur ausgetauscht, sondern durch ein sanftes Tweening von Weiß sichtbar gemacht. Wenn man weiterscrollt, werden die Produkte wieder über sechs Frames weiß, und das nächste Produkt wird sichtbar (siehe auch Abbildung 35).

Flash im Zusammenspiel

326 Zusammenarbeit mit anderen Programmen

326 Dreamweaver 4 und Flash
329 Fireworks und Flash
334 Vektorprogramme

336 Flash und RealVideo

339 Der RealPlayer 8
342 In fünf Schritten zum RealPlayer
355 Tipps zur Produktion
356 Interaktive Kommandos

362 Flash und QuickTime

362 Konvertierung von QuickTime-Video in Flash-Vektoren
362 Arbeitsmaterialien – ein Überblick
363 Umwandlung eines QuickTime Movies in einzelne Bitmaps
365 Nachbearbeitung im Grafikprogramm
368 Finish in Flash

370 Wiedergabe von Videosequenzen in Flash

370 Mithilfe des Movie-Explorers
372 Mithilfe der RAD-Tools
373 Der Flash-Import

374 Flash und JavaScript

374 Kommunikation zwischen Flashfiles in verschiedenen Frames
381 Flashfilm bildschirmfüllend – Randlos glücklich
385 Position einer Filmsequenz an Flash übergeben
388 Zusätzliche Browserfenster öffnen

396 Flash 5 JavaScript-Integration

400 Flash Player-Kontrollen in der Übersicht
406 Erweiterte Formularprüfungen in der Übersicht
416 Browser-Skripts für Flash in der Übersicht

422 Chatrooms in Flash

422 Flash-Chat

426 Erweiterung eines Flash-Movies mit dem Kommandozeileninterpreter

426 Schnittstellen zur Erweiterung
427 Der Kommandozeileninterpreter
429 Ein Programm selbst schreiben

Zusammenarbeit mit anderen Programmen

Dreamweaver, Fireworks und Vektorprogramme

Flash eignet sich ausgezeichnet auch zum Zusammenspiel mit anderen Programmen, besonders natürlich mit den übrigen Programmen der Macromedia-Produktpalette, auf die hier exemplarisch eingegangen werden soll.

Dreamweaver 4 und Flash

▸ Emrah Hircin, www.emrahnet.de

Auch Macromedia Dreamweaver ist ein Werkzeug, das keinem Flasher fehlen sollte. Nicht immer arbeitet man in Fullflash-Seiten. Es müssen Pop-ups gemacht werden, HTML-Startseiten, Framesets oder einfach Flash-Banner eingebaut werden. Sogar bei Fullflash-Seiten kann Dreamweavers WYSIWYG hilfreich sein.

In der neuen Version Dreamweaver 4 haben die Entwickler hauptsächlich das Konzept der Erweiterbarkeit von Dreamweaver noch weiter ausgereizt, sich aber auch bei Bedienungselementen (Look & Feel) an Flash angenähert.

Neue Objekte

Zwei interessante neue Objekte schmücken die Objektpalette von Dreamweaver 4.

Objekt für Flash-Text

Mit dem Objekt Insert Flash Text kann man – ohne Flash 5 installiert zu haben – Text als SWF generieren lassen und somit die erweiterten Möglichkeiten wie zum Beispiel die freie Schriftauswahl nutzen.

▲ Abbildung 1
Die neuen Flash-Objekte; Objekt für Flash-Text

Insert Flash Text
OK
Apply
Cancel
Help
HALLO! ICH BIN BLINDTEXT. BLINDTEXT ZU SEIN IST MANCHMAL GANZ SCHOEN LANGWEILIG ABER MAN WIRD IMMER GEBRAUCHT UND DAHER AUCH GUT BEZAHLT.

▲ Abbildung 2
Insert Flash Text

Sie können sogar ganze Textblöcke als SWF generieren lassen und die Ausrichtung bestimmen. Natürlich fehlt auch die Möglichkeit nicht, einen Link auf den Text zu legen. Zusätzlich kann man bei `MouseOver` eine Farbänderung bewirken.

Zuletzt muss man einen Dateinamen für das SWF vergeben. Nach Bestätigung der Eingaben generiert Dreamweaver das fertige SWF und fügt es gleich ein. Man kann nachträglich den Text noch ändern.

Objekt für Flash-Buttons

Ähnlich wie Flash-Text können Sie auch Flash-Buttons einfügen. Aus einer Reihe bestehender SWT-Dateien generiert Dreamweaver dann fertige Buttons, deren Texte Sie natürlich frei wählen können.

Später kann man Dreamweaver mit neuen Templates ausrüsten, und in absehbarer Zeit wird Macromedia auch sicherlich bekannt geben, welche Spezifikationen für die SWT-Templates gelten, so dass man selbst zur Tat schreiten kann.

Abbildung 3 ▶
Insert Flash Button
in Aktion

Asset-Manager

Der Asset-Manager dürfte vielen Flashern, die Dreamweaver nutzen, schnell ans Herz wachsen. Mit ihm kann man Bestandteile einer Site komfortabel verwalten und über sie verfügen. Auch die SWF-Dateien einer Site kann man so verwalten und einfach in das aktuelle Dokument ziehen.

So können Sie über mehrere Dateien einer Site hinweg den Überblick behalten.

JavaScript Integration Kit

Damit können Sie aus Flash heraus viele JavaScript-Routinen vereinfacht nutzen, darunter Cookies, Formularchecks und verfeinerte Steuerungsmethoden für Flash-Movies in HTML (Fast Forward etc.). Mehr dazu im Kapitel über JavaScript-Integration ab Seite 396.

◄ Abbildung 4
Das JavaScript Integration Kit stattet Dreamweaver mit Zusatzfunktionen für Flash aus.

Abbildung 5 ►
Export als PNG32 aus Fireworks

Fireworks und Flash

Fireworks eignet sich wegen einiger Eigenschaften insbesondere als Bildbearbeitungsprogramm für Flasher.

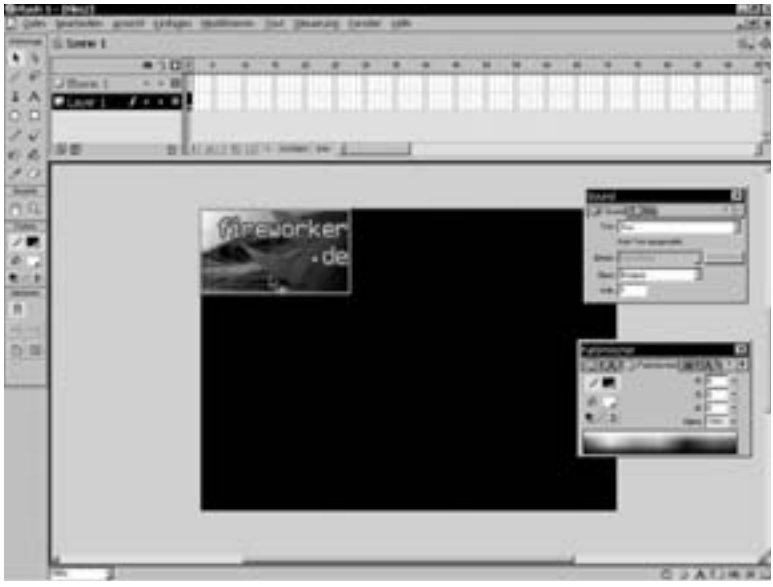
◄
Emrah Hircin, www.emrahnet.de

Export von Bitmaps für Flash

Fireworks erlaubt es, mit seinen Exportfunktionen PNG-Dateien mit Alphatransparenz zu exportieren. Die Alphatransparenz wird bekanntlich von Flash standardmäßig unterstützt und erlaubt daher den flexiblen Umgang mit Transparenzen.

Sie können jede Bitmap vor dem Import in Flash in Fireworks optimieren, um bestmögliche Ergebnisse in Flash zu erhalten. Um alle Transparenzen des Dokuments beizubehalten, müssen Sie sie als PNG32-Datei exportieren.

Abbildung 6 ▸
Exportiertes PNG in Flash inklusive Transparenzen



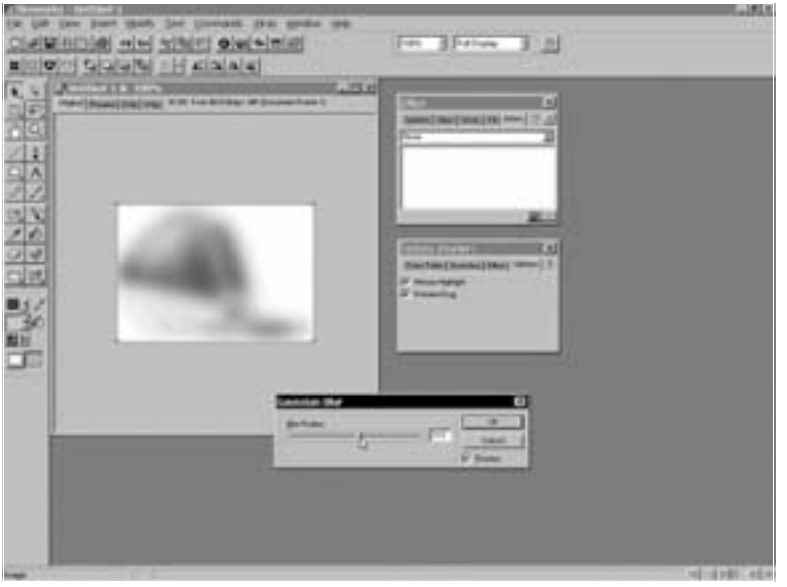
Import von Fireworks-PNG in Flash 5

In Flash 5 bietet Ihnen als Neuerung gegenüber der Version 4 an, die Fireworks-Quelldateien zu importieren. Fireworks speichert Dateien in einer speziell angepassten Version des PNG-Formats. Dieses enthält neben der reinen Bildinformation auch Informationen über Ebenen, Objekteigenschaften, Gruppierungen oder sonstige für Fireworks spezifische Merkmale.

In Flash bleiben dann sowohl Layer als auch Einzelobjekte wie z. B. Texte editierbar. In der Fireworks-Datei enthaltene Pfade sind den Bearbeitungswerkzeugen von Flash zugänglich, die in der Version 5 denen von Fireworks ähnlicher geworden sind.

An den Fireworks-Anwender im Flasher stellt diese neue Möglichkeit, sofern von ihr Gebrauch gemacht werden soll, auch neue Anforderungen. Die Fireworks-Datei sollte möglichst so organisiert sein, dass Elemente, die animiert werden sollen, schon in separaten Ebenen liegen, um in Flash keine unnötige Arbeit zu haben.

Bei Flash 4 musste man erst aus Fireworks in das Illustrator-Format exportieren, um dann diese Datei in Flash zu importieren.



◄ Abbildung 7
Der Weichzeichner von Fireworks in Aktion

In Flash 5 entfällt dieser Schritt. Natürlich ist es aber weiterhin als eine der weniger extravaganten, aber umso effektiveren Methoden möglich, einzelne Pfadobjekte aus Fireworks in Flash zu kopieren.

Effekte aus Fireworks in Flash importieren

Fireworks bietet mit seiner Kombination von Vektoren und Pixeln die Möglichkeit der Anwendung fotorealistischer Effekte auf Bitmaps, die Sie in Flash übernehmen können.

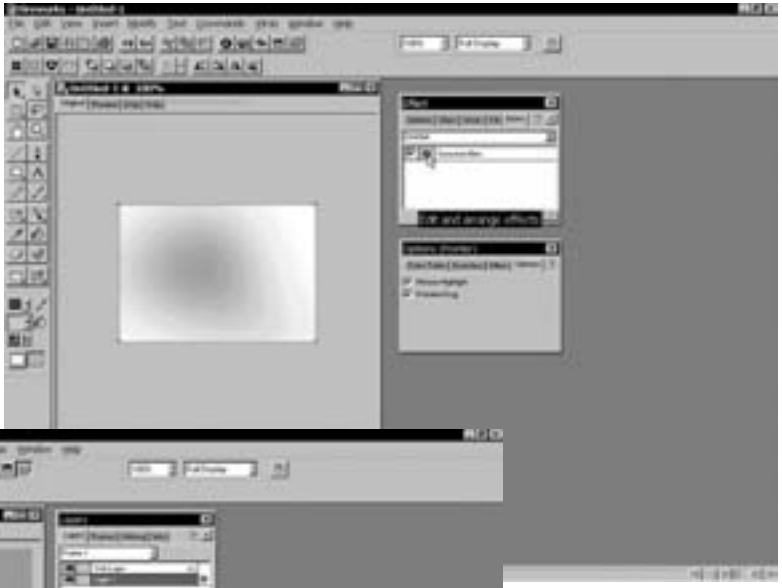
Blur-Effekt

Einen Blur-Effekt, mit dessen Hilfe Sie in Flash ein Objekt quasi in Rauch aufgehen lassen können, ist in Fireworks binnen weniger Minuten erstellt.

Sie müssen dazu einfach den zugehörigen Live-Effekt in Fireworks anwenden. Sie können den Blur-Effekt auch nachträglich noch variieren, was Ihnen eine große Flexibilität bietet.

Das Ergebnis können Sie einfach als PNG32 exportieren und auf einem der bekannten Wege in Flash importieren.

Abbildung 8 ▸
Fertig für den Export



◂ Abbildung 9
Effekte in Fireworks

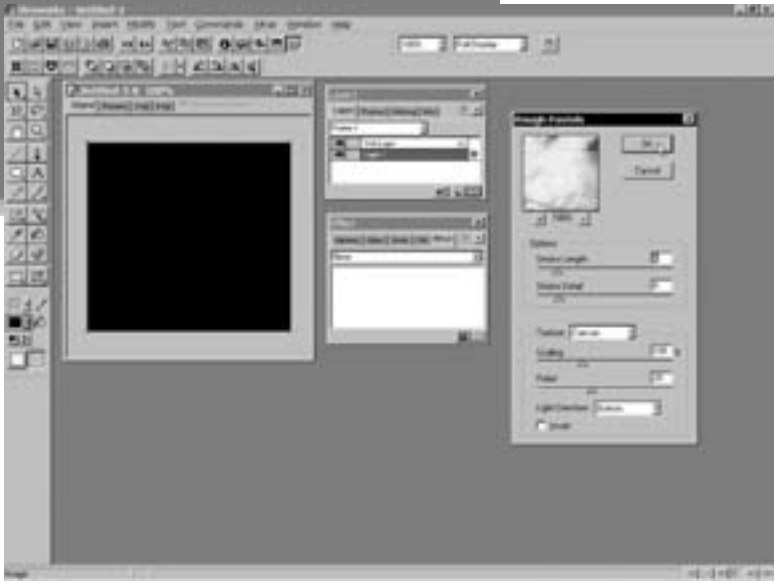


Abbildung 10 ▸
Anwendung eines Photoshop-Filters auf ein Bildobjekt in Fireworks

Drop Shadow, Emboss, Photoshop-Filter

Mit den Live-Effekten von Fireworks können Sie eine Vielzahl von Effekten erzielen, so zum Beispiel sehr gebräuchliche Effekte wie Emboss und Drop Shadow (siehe Abbildung 9).

Photoshop-Plug-ins können in Fireworks als Live-Effekte verwendet werden. So wird das Potenzial fotorealistischer Effekte ausgeschöpft, ohne die Flexibilität einzuschränken, da Sie den Effekt jederzeit mit einem Klick wegnehmen oder in seinem Ausmaß variieren können (Abb. 10).

SWF-Export in Fireworks

Fireworks verfügt über die Option, Animationen oder Einzelbilder als SWF zu exportieren. Sicherlich wird deswegen kein Flasher, der sein Werkzeug beherrscht, anfangen, mit Fireworks Animationen zu erstellen. Dazu wären die großartigen Möglichkeiten der Timeline von Flash viel zu schade.

Allerdings ist es in Einzelfällen nicht undenkbar, SWF-Dateien aus Fireworks zu exportieren, um sie in Flash per `LoadMovie` einzubinden.

Ausblick auf Fireworks 4

Für zukünftige Versionen von Fireworks ist zunächst eine weitere Angleichung der Bedienkonzepte beider Programme denkbar. Von Flash 4 auf Flash 5 wurden die meisten Bedienpaletten von Flash stark an die von Fireworks angepasst, so dass sich sowohl Flasher in Fireworks als auch Fireworker in Flash deutlich wohler fühlen dürften.

Wenn Macromedia es schaffen sollte, in Fireworks 4 einige unschöne Mängel in den Bildbearbeitungsfunktionen auch nur halbwegs auszubügeln, würde eine optimale Fireworks-/Flash-Achse entstehen, die auf eine effektive Arbeitsteilung zwischen beiden Programmen aufbauen würde.

So würde der gestalterische Teil in Fireworks stattfinden, die Animation und Interaktion in Flash.

Die Gewöhnung eines Flashers an Fireworks wird sicherlich in diesem Zuge stark erleichtert werden, sei es durch angeglichene Bedienung oder durch verbesserte Kooperationen.

Vektorprogramme

▶ Carlo Blatz, Gerald Marischka

Mit diesem Kapitel haben wir lange gekämpft. Sollen wir ein komplexes Kapitel zu allen vektorbasierten Grafikprogrammen machen, oder sollen wir sie nur nüchtern vorstellen? Wo fängt man an, und worauf beschränkt man sich? Wir haben uns entschlossen, gar nicht erst anzufangen. Um alleine die Möglichkeiten von FreeHand, Illustrator und CorelDraw vorzustellen, könnten wir ein ganzes Buch füllen.

Die meisten Grafiker haben mit Programmen wie FreeHand, Illustrator oder CorelDraw gelernt. Nur selten findet man Flasher, die beides können, und noch seltener Grafiker, die ihre Layouts bereits in Flash anlegen. Immer, wenn also ein komplexes Projekt entwickelt wird, werden die Grafiker nach ihren Layoutprogrammen gefragt. Dabei bleibt leider selten die Option der Auswahl des idealen Programms. Wenn ein Grafiker »Illustrator-Fan« ist, nützt es nichts, ihm von den Vorteilen des Flash-Imports von FreeHand-Dateien zu erzählen.

Wichtig ist nur, dass die Grafiken vom Kunden abgesegnet sind, bevor sie importiert werden. Eventuelle Änderungen müssen nach dem Import in Flash gemacht werden, somit vom Programmierer, sofern der Grafiker keine Flash-Erfahrung hat.

Man kann Adobe Illustrator direkt in Flash importieren. Besser eignet sich dafür aber das »Flashwriter«-Plug-in, das SWF-Dateien aus Illustrator erstellt.

Auch für CorelDraw gibt es ein Plug-in, das SWF-Dateien erstellt. (Die Ergebnisse waren in Flash 4-Zeiten mit am besten). Sowohl Vektoren, Text, Farben als auch Verläufe werden mit sehr guten Ergebnissen exportiert.

Seit Flash 5 FreeHand-Dateien importieren kann, hat sich die Frage fast erübrigt, welches Werkzeug sich am besten eignet. Vektoren und Verläufe hat FreeHand auch früher schon vernünftig in Flash-Dateien geschrieben, aber nun werden auch Texte im Block importiert, was früher aus allen Programmen ein Problem war.

Flash und RealVideo

Seit der Einführung des RealVideo-Servers G2 von RealNetworks unterstützt der RealPlayer auch Macromedia Flash. Was Sie mit Flash und dem RealPlayer bewerkstelligen können, erfahren Sie auf den folgenden Seiten.

▸ Detlef Randerath, www.detlef.de

Wer an Streaming-Media denkt, der denkt an RealNetworks, und das zu Recht.

Real Networks ist unumstritten der Marktführer im Bereich Streaming-Media im Internet. Die Serverlösungen von RealNetworks zeichnen sich durch Leistungsstärke und hohe Qualität aus. Als einer der Ersten entschloss sich RealNetworks dazu, dass Flash-Format zu unterstützen, und so wurde Flash 2 sehr früh in den RealPlayer integriert. Schon kurz nach der Einführung von Flash 2 konnte man Flash-Movies im RealPlayer wiedergeben.

Zur Wiedergabe der Streaming-Video- und Streaming-Audiodaten benötigt der Anwender im Internet den RealPlayer, eine Software, die in der Grundversion als RealPlayer Basic kostenlos von der RealNetworks Website unter http://www.realnetworks.com heruntergeladen werden kann.

Die RealPlayer Pro-Version bietet einiges an zusätzlichen Features und wird für einen Preis von US-Dollar 29.95 über die RealNetworks Website verkauft.

Wir werden uns auf den nächsten Seiten mit dem kostenlosen RealPlayer Basic beschäftigen, um unseren Flash-Exkurs in die Streaming-Media-Welt von RealNetworks zu realisieren.

Benötigte Software

Lassen Sie uns einen kurzen Blick auf die Software werfen, die wir benötigen, um Flash und den RealPlayer zusammenzubringen. Zuerst die gute Nachricht, wir benötigen keinen extra Server, auf dem wir einen RealServer installieren müssen. Zumindest solange wir nicht viel mit Videodaten

arbeiten und Hunderten oder gar Tausenden von Anwendern einen zeitgleichen Zugriff (Concurrent Streams) auf die Daten ermöglichen wollen.

RealNetworks bietet auf ihrer Website einen kostenlosen und voll funktionstüchtigen RealServer 7 mit 25 gleichzeitigen Streams für verschiedene Betriebssysteme zum Download an. Wer möchte, kann diese Server-Software auf seinem Server installieren und dazu verwenden, die Dateien, die wir im weiteren erstellen werden, im Internet anzubieten.

Natürlich benötigen wir Macromedia Flash. Mit Einführung des RealPlayer 8 hat RealNetworks endlich die neuen Flashformate in den RealPlayer integriert, so dass jetzt auch Flash 3- und Flash 4-Filme verwendet werden können.

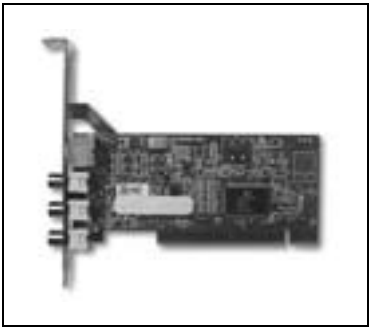
Aus diesem Grund konzentrieren wir uns in unserem Exkurs auf die RealPlayer Version 8, die einiges mehr kann als alle bisherigen RealPlayer-Vorgänger. Der RealPlayer 8 unterstützt neben dem Flash-Format noch eine große Anzahl weiterer nützlicher Formate, die wir uns gleich einmal genauer ansehen werden. Ein für uns sehr wichtiges Format ist die SMIL-Markup Language (Synchronised Multimedia Integration Language). **SMIL** (ausgesprochen wie engl. Smile) ist ein XML-Dialekt, der es Online-Autoren ermöglicht, Audio-, Video-, Text- und Bilddaten synchronisiert innerhalb des RealPlayers darzustellen.

Leider wird SMIL derzeit immer noch nicht von den führenden Browsern unterstützt, was leider dazu führt, dass SMIL ein eher selten eingesetztes Format ist. Innerhalb des RealPlayers ist SMIL für uns aber genau das richtige Mittel, um alle möglichen Formate in Zeitabhängigkeit voneinander darzustellen.

Um mit SMIL arbeiten zu können, benötigen wir einen normalen HTML-Editor. Ich persönlich verwende dafür am liebsten Macromedia Dreamweaver – oder, wenn's mal schnell gehen soll, den Windows-Editor. Wer sich so richtig ausführlich mit SMIL beschäftigen möchte und alle Features dieser sehr funktionellen Auszeichnungssprache kennen lernen möchte, dem empfehle ich einen Besuch auf der Website von Helio (http://www.helio.org), einer Non-Profit-Organisation mit Sitz in Frankreich, die auf ihren Seiten ein vorzügliches Tutorial zu SMIL bereithält. Wer möchte, kann das Tutorial auch als PDF-Datei herunterladen und offline studieren. Weitere nützliche Links im Internet in Bezug auf SMIL finden Sie am Ende dieses Kapitels.

Für die Produktion der verwendeten Bilddateien eignen sich bestens Macromedia Fireworks sowie Adobe Photoshop. Damit wir auch etwas Streaming-Video produzieren können, verwenden wir den RealProducer Basic von RealNetworks. Mit dem RealProducer Basic kann man übrigens

Abbildung 1 ▶
Osprey 100 PCI-Karte



neben RealVideo- auch RealAudio-Dateien produzieren und nachbearbeiten.

Hier also noch einmal in der Übersicht, was wir benötigen:

Programm	Verwendung
Macromedia Flash	für die Flash-Movies
Fireworks oder Photoshop	für Bilddateien
Dreamweaver oder Edit	für die SMIL-Bearbeitung
RealProducer	für die Produktion von RealVideo-Dateien
RealPlayer 8 Basic	für die Wiedergabe unserer Dateien

Videoquelle

Für die Produktion der Videofilme, auf die wir später auch noch kurz eingehen wollen, benötigen wir zusätzliche Hardware, die es ermöglicht, eine Video-/Audioquelle an unseren PC anzuschließen und das vorhandene Signal mittels der RealProducer Basic-Software zu digitalisieren.

Als Videoquelle eignen sich Videorekorder oder Videokameras, die heute über standardisierte Anschlüsse verfügen. Für die eigentliche Digitalisierung verwenden wir eine PC-Capture-Karte, wie zum Beispiel die **Osprey 100**, die von RealNetworks mit vielen Serverprodukten im Bundel verkauft werden. Die Osprey 100 PCI-Karte kann für US-Dollar 200 über die Realnetworks Website unter www.realnetworks.com online erworben werden.

Die Osprey 100-Karte ist leicht zu installieren und wird vom RealProducer Basic sofort nach der Installation erkannt und unterstützt. Ein

◀ **Abbildung 2**
Belkin Videobus-Adapter

Nachteil von PCI-Karten ist allerdings, dass diese in den PC eingebaut werden müssen, was nicht unbedingt jedermanns Sache ist.

Einfacher lassen sich da externe Schnittstellen in Betrieb nehmen, die zum Beispiel den USB-Port verwenden. Hier bietet sich der **Belkin Videobus-Adapter** an, der bei der Installation lediglich am USB-Port angeschlossen wird. Die Installation läuft denkbar einfach ab, Software installieren – Videobus in den USB-Port stecken – Verbindung mit dem Videorekorder herstellen, und schon kann es los gehen mit der Produktion.

Der Belkin Videobus kann im Internet unter www.belkin.com für die Systeme Windows 98/2000/NT sowie für Apple Macintosh für einen Preis von US-Dollar 100 online erworben werden. Der Vorteil der beiden hier genannten Systeme ist der, dass beide Systeme vom RealProducer Basic unterstützt werden. Wer sich mit dem Thema weiter auseinandersetzt, wird feststellen, dass am Markt eine Vielzahl von Systemen erhältlich ist, die Video- und Audiodateien in einer sehr hohen Qualität erzeugen können, leider wird nicht jedes System von der RealProducer-Software unterstützt.

Vor dem Kauf sollte man versuchen einen Test durchzuführen oder sich auf der RealNetworks Website informieren, ob das gewünschte System unterstützt wird.

Der RealPlayer 8

Die neueste Version des RealPlayers wurde stark gegenüber den Vorgängerversionen verbessert und bietet viele neue Features und natürlich eine noch bessere Audio- und Videowiedergabe. Was aber für uns besonders interessant ist, ist die Tatsache, dass der RealPlayer 8 nun endlich auch

Abbildung 3 ▸
Die RealPlayer 8-Oberfläche

das Flash 4-Format unterstützt. Alle vorhergegangenen RealPlayer (5.0, G2 und 7) unterstützten das ältere Flash 2-Format.

Der RealPlayer 8 bietet eine Vielzahl neuer Features und unterstützt jetzt auch neben einigen neuen Formaten die Formate Flash 3 und Flash 4.

Unterstützte Formate im RealPlayer

Der RealPlayer 5.0, RealPlayer G2 und der RealPlayer 7.0 waren bislang in der Lage, Flash 2-Movies anzuzeigen. Erst die kürzlich neu erschienenen Versionen RealPlayer 8 Basic und RealPlayer 8 Plus unterstützen das Flash 3- und das Flash 4-Format. Durch die erweiterte Funktionalität ist der RealPlayer nun auch in der Lage, interaktive Kommandos aus dem Flash-Film weiter zu verarbeiten, was die Arbeit mit SMIL einerseits auf ein Minimum reduziert und uns auch die Möglichkeit gibt, mehr mit Flash selbst zu arbeiten.

Wer abwärtskompatibel für die älteren RealPlayer produzieren möchte, kann hierfür natürlich auch Flash 4 und Flash 5 verwenden, muss allerdings darauf achten, dass diese Player nur Flash 2 unterstützen, und dementsprechend die erstellten Filme als Flash 2-Filme exportieren, was

dazu führt, dass viele der neuen Features in Flash 4 und Flash 5 verloren gehen.

Das im RealPlayer verwendete Flash-Plug-in ist leider nicht das gleiche Plug-in, das auch in den Webbrowsern zum Einsatz kommt. Dementsprechend steht auch nicht die volle Funktionalität von Flash im RealPlayer zur Verfügung. Ein Beispiel hierfür ist die Unterstützung von Mouse-Events. Während das Flash-Browser-Plug-in Mouse-Events wahrnimmt, zum Beispiel wenn ein Button innerhalb eines Flash-Films gedrückt wird und mit gedrückt gehaltener Maustaste der Mauszeiger außerhalb des Flash-Films bewegt und erst dann losgelassen wird, nimmt dies das Browser-Plug-in wahr und reagiert mit dem Button innerhalb des Flash-Films. Dem RealPlayer-Flash-Plug-in fehlt diese Funktionalität vollkommen. In diesem Fall reagiert dann auch der Flash-Film anders und aktualisiert den Button nicht, wenn die Maustaste losgelassen wird.

Unterstützte Dateiformate

Dateiendungen	Dateityp
RM, RA, RAM	RealAudio-Streams und RealVideo-Streams
RT	RealText gestreamtes Textformat
RP	RealPix gestreamte PNG-, GIF- und JPG-Bilder
PNG, GIF, JPG	PNG-, GIF- und JPG-Bilder und animierte GIF-Bilder
MP3	Mpeg Layer 3 (Audioformat)
SWF	RealSystem G2 mit Macromedia Flash
SMIL, SMI	Synchronised Multimedia Integration Language
WAV, AIFF	Audioformate (aber keine älteren Codecs)
MPG, MPEG	MPEG Layer 1 Video- und Layer 2 Audioformate
AVI	Microsoft Audio-/Videoformate
ASF	NetShow Dateiformat
MID, MIDI, RMI	MIDI Audioformate

RealNetworks hat mit der Plug-in-Funktionalität ermöglicht, durch Hinzufügen weiterer Plug-ins das Leistungsspektrum des RealPlayers zu erweitern. Das gibt vielen Entwicklern die Möglichkeit, ihre Dateiformate mittels eines Plug-ins in den RealPlayer zu integrieren. Stößt der RealPlayer beim Laden einer Datei auf ein Format, das er noch nicht unterstützt, versucht er das passende Plug-in zu finden und installiert dieses selbstständig, wenn dieses zur Verfügung steht.

▲ Abbildung 4
Der offizielle Download-Button für den RealPlayer

Lassen Sie uns nun unseren kleinen Exkurs in die Welt des Streaming-Media beginnen, indem wir uns etwas näher mit dem RealPlayer beschäftigen.

Bei diesem Button können Sie sicher sein, dass Sie die aktuelle Version des RealPlayers von der RealNetworks-Website herunterladen können. Wer mag, kann seine Website für die Distribution des RealPlayers registrieren lassen und den Besuchern seiner Website einen direkten Download ermöglichen.

In fünf Schritten zum RealPlayer

Los geht's, nach der Installation des RealPlayers können Sie Ihre ersten Erfahrungen mit dem RealPlayer machen. Bestens geeignet für einen ersten Ausflug ins Land des Streaming-Media ist das deutschsprachige Portal von RealNetworks – dem RealGuide, zu finden unter http://www.realguide.de.

Wie entwickeln wir nun unsere Inhalte für den RealPlayer? Natürlich in Flash.
Wir beginnen mit einem Flash-Menü, das uns den Zugriff auf verschiedene Dateiformate aus dem RealPlayer heraus ermöglichen wird. Kopieren Sie hierzu den Ordner Übungen aus dem Verzeichnis RealVideo auf Ihre Festplatte.

Das Flash-Movie

Öffnen Sie die Flash-Datei Menue.fla im Ordner Übungen. Der Flash-Film ist bereits aufgebaut und enthält drei Buttons, die wir im Folgenden noch mit verschiedenen ActionScripts versehen werden, um den Zugriff auf die gewünschten Dateien zu ermöglichen.

Mit dem Button Der Boxenstopp wollen wir einen einfachen Real-Media-Film in einem neuen RealPlayer-Fenster öffnen. Etwas später werden wir uns mit SMIL etwas weiter beschäftigen, um in unserem Menü mittels SMIL einen weiteren RealVideo-Film zu integrieren, der zeitgleich mit unserem Menü abgespielt wird.

Aber nun wollen wir uns zunächst mit dem Aufruf eines neuen Real-Video-Films beschäftigen, dazu sehen wir uns die beiden folgenden GetURL-Kommandos an.
Dieses Kommando ruft den neuen RealVideo-Film in einem neuen Fenster mittels des Parameters `_new` auf:

◀ **Abbildung 5**
Unser Flash-Übungsmenü im RealPlayer 8

```
command:openwindow(_new,rtsp://ms.mediacluster.d.de.europop.net:554/
oschersleben/probike_edition.rm, autosize=true, ontopwhileplaying=true)
```

Dieses Kommando nutzt zum Aufruf des neuen RealVideo-Films den Parameter `_blank`:

```
command:openwindow(_blank,rtsp://ms.mediacluster.d.de.europop.net:554/
oschersleben/probike_edition.rm, autosize=true, ontopwhileplaying=true)
```

Wer mag, kann den RealVideo-Film auch in das aktuelle Fenster laden, dazu eignen sich die folgenden beiden Kommandos.

```
command:openwindow(_self,rtsp://ms.mediacluster.d.de.europop.net:554/
oschersleben/probike_edition.rm)
command:openwindow(_current,rtsp://ms.mediacluster.d.de.europop.net
:554/oschersleben/probike_edition.rm)
```

Nachdem wir nun in der Lage sind, weitere RealVideo-Filme in neuen Player-Fenstern zu öffnen, möchten wir unser Menü dazu befähigen, eine HTML-Seite in einem Browserfenster zu öffnen. Bislang war das mit wesentlich mehr Aufwand verbunden als in der neuen Version des RealPlayers.

In der neuen Version werden die GetURL-Kommandos aus dem Flash-Film gleich an den Browser weitergeleitet. Das macht die Handhabung dieser Kommandos sehr einfach, da wir keine spezielle Syntax für den RealPlayer hierzu einhalten müssen.

Abbildung 6 ▸
GetURL

Öffnen Sie in unserem Flash-Menü den Button Der Zieldurchlauf und klicken auf den Menüreiter Aktionen. Weisen Sie dem Button wie in der folgenden Abbildung gezeigt die Aktion GetURL mit einer gültigen HTTP-Adresse zu. Sie können natürlich auch eine lokale Adresse zum Testen eingeben.

Mit dem dritten Button möchten wir nun in unseren aktuellen RealPlayer 8 eine neue SMIL-Datei laden, die dann unser Menü im aktuellen Player ersetzen soll.

Hier setzen wir wieder auf die neuen Features des RealPlayer 8 und weisen dem Button in unserem Flash-Menü lediglich ein GetURL-Kommando wie in der unten gezeigten Abbildung zu.

Analog zum Aufruf einer HTML-Datei werden auch die GetURL-Kommandos bearbeitet, die andere Dateiformate beinhalten. In diesem Fall wird eine SMIL-Datei aufgerufen, für deren Wiedergabe wiederum der RealPlayer zuständig ist.

Damit wir uns von der Funktionalität unseres Flash-Menüs überzeugen können, speichern Sie dieses auf Ihre Festplatte unter einem neuen Namen und exportieren anschließend den fertigen Film in den Ordner Übungen.

Wie testet man nun den neu erstellten Flash-Film? Die einfachste und schnellste Methode besteht darin, den Flash-Film (.swf) per Drag and Drop in den bereits geöffneten RealPlayer zu ziehen. Dieser erkennt das Dateiformat und beginnt sofort mit der Wiedergabe.

Beachten Sie bitte bei der Produktion, dass Sie den Flash-Film, den Sie gerade im RealPlayer testen, nicht auf Ihrer Festplatte überschreiben können. Flash erkennt, dass der gewünschte Flash-Film von einem anderen

Programm (in diesem Fall dem RealPlayer) benutzt wird, und weigert sich, diesen Film zu überschreiben.

Entweder Sie erstellen eine Kopie des Flash-Films, den Sie gerade testen möchten, oder Sie öffnen und schließen den RealPlayer für jeden neuen Test. Diese Methode stellt sicher, dass Sie immer den letzten neuen Flash-Film in einem neuen sauberen RealPlayer testen können. Auch der RealPlayer kann sich an komplexen Flash-Filmen und großen SMIL-Dateien einmal verschlucken, ein neuer RealPlayer kommt uns dann für den nächsten Test sehr gelegen.

Wer auf Nummer sicher gehen will, der installiert auf einem Server den kostenlosen RealServer Basic, der auf der Website von RealNetworks unter Products/Free Products for Evaluation/Real Server Basic zum Download angeboten wird. Hierbei handelt es sich um einen funktionstüchtigen RealServer, der 25 gleichzeitige (concurrent) Streams zulässt. Der RealServer Basic steht für die folgenden Betriebssysteme zum Download bereit:

- Windows 2000 Intel
- Windows NT Intel
- Linux 2.2 (libc6)
- Sun SunOS 5.6 (Solaris 2.6)
- Sun SunOS 5.7 (Solaris 2.7)
- Sun SunOS 5.8 (Solaris 2.8)
- SGI Irix 6.2, SGI Irix 6.5
- SCO Unixware 7.0.1
- SCO Unixware 7.1.0
- SCO Unixware 7.1.1
- FreeBSD 3
- HP-UX11

Nach der Installation des RealServer Basic können Sie Ihre Dateien im Internet sozusagen in der »Echtumgebung« testen. Nutzen Sie den »Bandwith Tuner«, den wir etwas später kennen lernen werden, um Ihre Flash-Filme für die Wiedegabe im Internet zu optimieren.

Das Flash-Menü ist fertig – schauen wir uns nun die weitere Produktion an.

Abbildung 7 ▸
Das Startfenster mit Recording Wizard des RealProducer Basic 8

Select a Recording Wizard for the media clip that you want to create.
Record From File
This option allows you to create a media clip from existing audio or video files located on your hard drive.
Record From Media Device
This option allows you to capture a media clip from a media device such as a Microphone, PC Camera, CD Player, or VCR connected to your computer.
Live Broadcast
This option allows you to broadcast a live media stream from your computer to a RealNetworks RealServer.
Don't Use Recording Wizards
OK
Cancel
Clip Information
Title
Author
Copyright
©2000
Description
Keywords
Recording Controls
28K Modem
56K Modem
Single ISDN
Dual ISDN
Corporate LAN
Closed

Der Videostream

Im folgenden Abschnitt beschäftigen wir uns kurz mit der Erstellung von RealVideo-Filmen, die wir dann durch unser Flash-Menü im RealPlayer aufrufen und wiedergeben werden.

Wie bereits erwähnt, benötigen wir zur Produktion von RealVideo-Filmen zusätzliche Hardware, eine Video-Capture-Karte für den Einbau in den PC oder eine externe Einheit wie zum Beispiel der Belkin USB-Videobus. Als Videoquelle verwenden wir entweder einen Videorekorder oder eine Videokamera. Wichtig ist, dass alle Verbindungen zwischen Computer und Videoquelle mit dafür geeigneten Kabeln hergestellt werden, um Beschädigungen am PC auszuschließen.

Nachdem die Hardware (Capture-Karte oder USB-Adapter) installiert sind, können wir die Software einrichten. Starten Sie nun aus dem Ordner Real/Software das Programm RealProducer_8.0_Setup.exe, um den RealProducer Basic zu installieren. Folgen Sie den Installationsanweisungen, und starten Sie anschließend den RealProducer Basic.

Der RealProducer ist ein sehr anwenderfreundliches Programm, das die wichtigsten Schritte bei der Produktion des Audio-/Videomaterials über den so genannten Recording Wizard ermöglicht.

◀ **Abbildung 8**
Der Wizard

Durch den Wizard wird sichergestellt, dass der Anwender alle zur Produktion notwendigen Schritte berücksichtigt und mit den Grundeinstellungen des Programms vertraut gemacht wird.

Ein Wort an dieser Stelle zum RealProducer Basic. Der einzige Unterschied zum großen Bruder, dem RealProducer Pro, ist eine Einschränkung im Bereich »Target Audience«. Hier können im Gegensatz zur Pro-Version nur zwei Bandbreiten ausgewählt werden. Für unseren ersten Exkurs in die Welt des RealPlayers genügt uns der RealProducer Basic vollkommen. Für professionelle Produktionen empfehle ich aber den RealProducer Pro zu verwenden.

Der RealProducer Basic – hier können Sie Ihre Einstellungen vornehmen.

Der Recording Wizard hilft uns dabei, die nötigen Einstellungen für die Aufnahme zu machen. Die Clip-Informationen werden im Audio-/Videoclip gespeichert und können im RealPlayer dargestellt werden.

Im Bereich RealMedia Settings wählen wir MULTI-RATE SURESTREAM aus. Dieses Format fügt die ausgewählten Bandbreiten in einer Datei zusammen. Das bedeutet in dem hier gezeigten Fall, dass Anwender von

56-K-Modems und ISDN-Anwender die gleiche Datei benutzen. Zusätzlich stellen wir hier ein, dass mit dem Audioformat Voice Only nur Sprache und keine Hintergrundmusik eingestellt wird und die Videoqualität mit Sharpest Image Video die beste Videoqualität ist.

Im linken Vorschaufenster des RealProducer Basic können wir nach dem Starten der Videoquelle das Vorschaubild sehen, das den RealProducer erhält. Im rechten Vorschaubild sehen wir während der Aufnahme das komprimierte Videobild, das der RealProducer auf die Festplatte speichern wird.

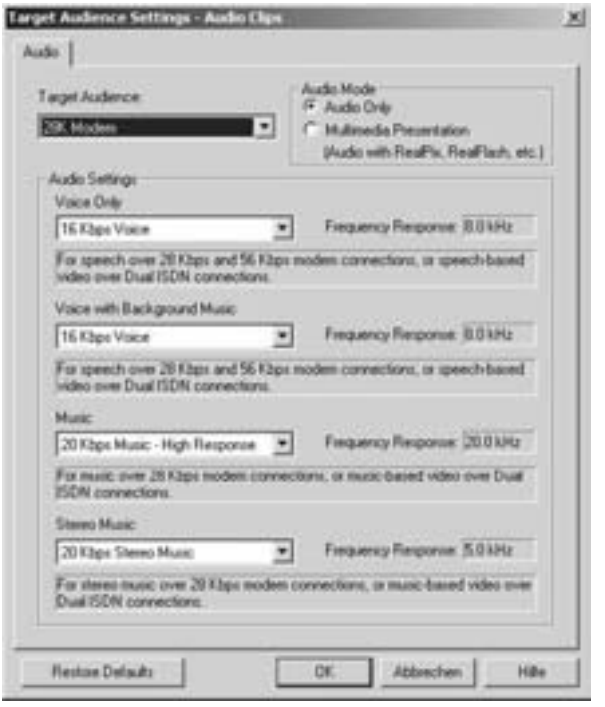
Hier können Sie mit der Qualität für Aufzeichnung und Ton spielen, um die optimale Qualität aus Ihrer Videoquelle herauszuholen. Hier gilt: »Übung macht den Meister.« Es kommt ganz darauf an, was für eine Videoquelle zur Verfügung steht und in welcher Qualität der Videofilm vorliegt. Schnelle Schwenks und grellbunte Aufnahmen liegen dem RealProducer nicht so sehr. Wer extra für die Anwendung im Internet Videomaterial aufnimmt, sollte auf Schwenks und schnelle Zooms verzichten. Lieber mehr Schnitte in den Film einfügen als lange Kamerafahrten.

Der Audiostream

Video ist nicht alles, es gibt auch noch RealAudio – für den guten Ton im Internet. Besonders die Wiedergabe von Audiodateien macht den RealPlayer so beliebt bei den Anwendern. In Verbindung mit SMIL können beeindruckende Präsentationen erstellt werden. RealNetworks bietet für die Produktion solcher Präsentationen, die Grafiken, Audio- und Videodateien sowie Texte beinhalten können, spezielle Tools an. RealSlideShow und RealPresenter sind neben dem RealProducer und dem eigentlichen RealServer die wichtigsten Werkzeuge, die RealNetworks derzeit anbietet.

Für die Produktion von Audiodateien kommen, wie schon gehabt, Videorekorder und Videokameras als auch Tonbandgeräte und CD-Player in Frage. Die Audioquelle wird an die Soundkarte des PCs angeschlossen, die dann im RealProducer als Audioquelle eingestellt werden kann. Achten Sie bitte auf die richtige Verkabelung, wenn Sie CD-Player oder Videogeräte anschließen. Natürlich können auch Fernseh- oder Radiogeräte als Quelle dienen.

Im Menü Options·Target Audience Settings for RealAudio Clips können Sie verschiedene Einstellungen für die Aufnahme von Audiodaten eingeben.


Target Audience Settings - Audio Clips
Audio
Target Audience:
28K Modem
Audio Mode
Audio Only
Multimedia Presentation
(Audio with RealPix, RealFlash, etc.)
Audio Settings:
Voice Only
16 Kbps Voice
Frequency Response: 8.0 kHz
For speech over 28 Kbps and 56 Kbps modem connections, or speech-based video over Dual ISDN connections.
Voice with Background Music
16 Kbps Voice
Frequency Response: 8.0 kHz
For speech over 28 Kbps and 56 Kbps modem connections, or speech-based video over Dual ISDN connections.
Music
20 Kbps Music - High Response
Frequency Response: 20.0 kHz
For music over 28 Kbps modem connections, or music-based video over Dual ISDN connections.
Stereo Music
20 Kbps Stereo Music
Frequency Response: 15.0 kHz
For stereo music over 28 Kbps modem connections, or music-based video over Dual ISDN connections.
Restore Defaults
OK
Abbrechen
Hilfe

◄ **Abbildung 9**
Options – Target Audience Settings for RealAudio Clips

In der Abbildung sehen wir die Standardeinstellungen für die Produktion von RealAudio-Dateien (.ra). Die Einstellungen werden in einem Textfenster unterhalb der Drop-down-Menüs zusätzlich erklärt und unterstützen so den Anwender bei seinen Einstellungen.

Auch hier gilt es, sich mit den verschiedenen Einstellungen und den aufgenommenen Ergebnissen vertraut zu machen und mit verschiedenen Quellen zu arbeiten. Für Musik und Sprache eignen sich hervorragend CD-Player und mobile Minidisk-Rekorder, die qualitativ hochwertige Ton- und Sprachaufnahmen ermöglichen und schon recht günstig im Handel zu haben sind.

Die von Ihnen produzierten RealAudio-Dateien eignen sich bestens zur Synchronisation von Animationen und Bildpräsentationen. Achten Sie bei der Produktion von Flash-Filmen für den RealPlayer darauf, dass diese keine Sounddateien enthalten. Hintergrundmusik und Tonsynchronisation wie zum Beispiel ein Sprechertext lassen sich im RealPlayer mittels SMIL optimal synchronisieren und können gestreamt werden. Das bedeutet, dass die Audiodatei während der Wiedergabe noch übertragen

wird und die eigentliche Wiedergabe durch dieses Feature ohne große Zeitverzögerung beginnen kann.

RealAudio-Dateien können ohne weiteres auch ohne SMIL im RealPlayer wiedergegeben werden, mit SMIL lassen sich aber Hintergrundanimationen und RealAudio-Dateien ohne großen Aufwand perfekt synchronisieren.

Wer sich weiter mit der Produktion für den RealPlayer beschäftigen möchte, dem empfehle ich das Authoring Kit 8 unter folgender Adresse herunterzuladen:

http://www.realnetworks.com/devzone/downlds/authkit/index.html

Hier werden alle wichtigen Komponenten sehr ausführlich beschrieben und eine Vielzahl von nützlichen Links zu weiteren Informationen auf der Website von RealNetworks angeboten.

SMIL – Schnelleinstieg

Damit alle Komponenten optimal zueinander finden, werden wir an dieser Stelle einen kleinen Exkurs in die Möglichkeiten von SMIL (Synchronised Multimedia Integration Language) machen.

Die Entwickler bei RealNetworks haben sich einiges bei der Integration von SMIL gedacht. Die Syntax ist von HTML abgeleitet und ähnlich leicht zu handhaben. Der Quellcode ist leicht zu lesen und kann dementsprechend einfach modifiziert und erweitert werden. Damit nun alle bisher produzierten Komponenten zusammenfinden und im RealPlayer gemeinsam wiedergegeben werden können, schreiben wir eine SMIL-Datei. Dazu verwenden wir entweder einen normalen Texteditor wie zum Beispiel Edit unter Windows oder einen geeigneten HTML-Editor wie beispielsweise Macromedia Dreamweaver.

Als Erstes legen wir eine neue Datei an und speichern diese unter dem Namen menue2.smi im Ordner Übungen auf Ihrer Festplatte. Die Dateiendungen für SMIL-Dateien lauten .smi und .smil analog zu den Dateiendungen von HTML-Seiten.

SMIL-Code für die Datei menue2.smi

```
<smil>
<head>
<meta name="title" content="Flash im RealPlayer" />
<meta name="author" content="Detlef Randerath" />
<meta name="copyright" content
="EUROPOP(R) AG" />
```

```
<meta name="abstract" content="Detlef Randerath erklärt SMIL"/>
<!-- Hier wird das Layout der Seite festgelegt -->
<layout>
   <region id="background" top="0" left="0" width="550"
height="300"/>
   <region id="video" top="50" left="285" width="260" height"90"/>
</layout>
</head>
<body>
<!-- in diesem Bereich werden die Dateien verknüpft und synchroni-
siert -->
<par>
   <animation src="menue.swf"
region="background" fill="freeze" />
   <video src="boxenstop.rm" region="video" />
</par>
</body>
</smil>
```

Der Aufbau der SMIL-Datei ist dem einer HTML-Datei nachempfunden, da es sich bei SMIL um einen so genannten XML-Dialekt handelt. Im Kopf der Datei werden Informationen zum Dateityp, Autor, dem Copyright und dem Inhalt der Datei gemacht. Anschließend wird im Kopfbereich der Datei das Layout der verschiedenen Komponenten definiert.

Der Tag `Region` legt fest, welchen Bereich die einzelnen Komponenten später nutzen werden. Zusätzlich erhält jede Region eine ID, auf die später im Body der SMIL-Datei eingegangen wird.

Im eigentlichen Body der Datei, wie wir ihn von HTML-Seiten kennen, werden die Dateien verknüpft und mit den im Kopf angegebenen IDs verknüpft. Der RealPlayer kennt vom Kopf der Datei das Layout und nutzt nun die IDs dazu, die entsprechenden Komponenten in den angegebenen Regionen darzustellen.

Dazu bietet SMIL sehr interessante Möglichkeiten. Zum parallelen Abspielen von Dateien empfiehlt sich die Verwendung des <par></par>-Tags, den wir in unserem Listing oben verwendet haben. Durch das Einbetten des Flash-Films und des Boxenstop.rm-Films werden diese beiden Dateien gleichzeitig (parallel) wiedergegeben.

Wir haben im oben gezeigten SMIL-Code zusätzlich den Parameter `fill="freeze"` verwendet, um die Laufzeit unseres Flash-Films auf die Laufzeit des synchron abgespielten Filmes einzustellen. Durch diesen

Parameter bleibt es uns erspart, den Flash-Film auf mehrere Minuten Laufzeit zu strecken, damit beide Dateien die gleiche Lauflänge erhalten.

SMIL bietet eine Vielzahl von interessanten Features und nützlichen Parametern, um ansprechende Animationen aus einfachen Bildern und Texten als auch aus Audio- und Videodateien zusammenzustellen.

Der Tag `<seq></seq>` zum Beispiel erlaubt die zeitgesteuerte sequenzielle Wiedergabe von Inhalten. Hierbei kann wirklich sekundengenau gearbeitet werden. Mit dem `<switch></switch>`-Tag kann man Abfragen generieren, die zum Beispiel einen englischen Text einblenden, wenn ein englisches Betriebssystem vorgefunden wird, oder eine kleinere Grafik laden, wenn die System-Bitrate zu niedrig ist für hochauflösende Grafiken.

Ich möchte an dieser Stelle nochmals auf das SMIL-Tutorial von Helio hinweisen (http://www.helio.org), das einen sehr schnellen Start mit SMIL ermöglicht und sehr ausführlich auf die vielen Parameter eingeht, sowie auf das Authoring Kit 8 von RealNetworks.

Wer professionelle Inhalte für den RealPlayer erstellen möchte, sollte sich als Entwickler bei RealNetworks registrieren. Der erste Einstiegslevel als Entwickler ist kostenlos und bietet Zugriff auf sehr interessante und hilfreiche Informationen rund um das Thema Streaming-Media.

Wer RealVideo-Daten live im Internet streamen möchte, muss sich zudem mit dem RealServer beschäftigen, denn ohne einen RealServer gibt es kein aktives Streaming. Wenn Sie sich lieber nicht mit dem Server auseinander setzen möchten und dennoch live ins Internet übertragen wollen, wenden Sie sich an einen Dienstleister wie zum Beispiel die EUROPOP AG in Düsseldorf (www.europop.net), die ihre RealServer und die entsprechenden Backbone-Anbindungen vermieten und Hilfestellung bei Projekten anbieten.

Zum Abschluss der SMIL-Erklärung sehen wir uns noch den SMIL-Code für die zweite SMIL-Datei an, die unser Flash-Menü in den RealPlayer laden soll.

SMIL-Code für die Datei infos.smi

```
<smil>
<head>
<meta name="title" content="Flash im RealPlayer"/>
<meta name="author" content="Detlef Randerath"/>
<meta name="copyright"
content="EUROPOP(R) AG"/>
<meta name="abstract" content="Detlef Randerath erklärt SMIL"/>
<layout>
```

```
<root-layout background-color="black" width="550" height="300"/>
<region id="Bild" left="120" top="30" width="320" height="240"/>
</layout>

</head>
<body>
<par>
   <img src="images/renn1.gif" region="Bild" width="320"
height="240" fill="freeze">
   <anchor href="menue.smi" />
   </img>

</par>
</body>
</smil>
```

In diesem Beispiel wird der Hintergrundbereich mit der Farbe Schwarz gefüllt und ein Bild mittig darüber positioniert. Damit wir wieder zu unserem Menü zurückkehren können, fügen wir innerhalb des Image Tags einen <anchor>-Tag ein, der einen ganz normalen HTML-HREF nutzt, um die menue.smi-Datei wieder im gleichen Fenster aufzurufen.

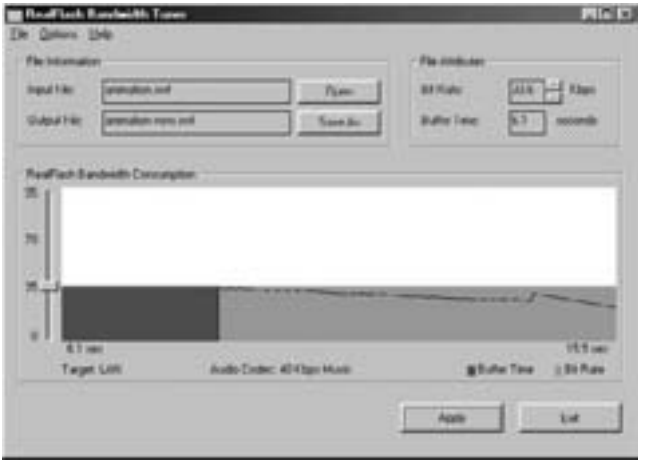
SMIL-Dateien können Sie zum Testen einfach per Drag and Drop in den RealPlayer ziehen, um diese wiederzugeben. Oder Sie laden Ihre SMIL-Dateien über den Dialog Datei·Datei öffnen in den RealPlayer.

Bandwith Tuner

Bevor alles online gehen kann, optimieren wir den erstellten Flash-Film mit dem Bandwith Tuner. Dieses kleine Programm, das von den Entwicklern immer noch gerne Afterburner genannt wird, dient der Optimierung der Streamfähigkeit des Flash-Films.

Wenn Sie wie in unserem Beispiel verschiedene Mediatypen (Flash und RealVideo) innerhalb einer SMIL-Datei an den RealPlayer übergeben, müssen die entsprechenden Daten (also die Flash-Datei und der Real-Video-Film) zum Player gestreamt werden. Das bedeutet, dass mit dem ersten Bild begonnen wird und, sobald diese Daten im RealPlayer ankommen, auch sofort dargestellt werden. Der RealPlayer ist so aufgebaut, dass er beim Aufrufen einer Datei zunächst genügend Daten in seinen Pufferspeicher lädt, bevor die eigentliche Wiedergabe beginnt. Die Größe dieses Puffers kann vom Anwender eingestellt werden.

Abbildung 10 ▶
Bandwith Tuner



Warum optimieren wir den Flash-Film? Der Bandwith Tuner erlaubt uns, das Streaming-Verhalten des Flash-Films zu verändern bzw. zu optimieren. In unserem Beispiel wird neben dem Flash-Menü noch ein RealVideo-Film gleichzeitig mit zum RealPlayer übertragen. Damit beide Dateien nun möglichst zeitgleich wiedergegeben werden können, stellen wir die Bitrate des Flash-Films so ein, dass sie der Übertragung (dem Streaming) des RealVideo-Films nicht »im Wege« steht.

Durch die Veränderung der Bitrate unseres Flash-Films verändert sich die Bufferzeit, die festlegt, wie lange es dauert, bis der Flash-Film im RealPlayer dargestellt wird. Dabei gilt es zu beachten: je höher die Bitrate, desto kürzer ist die Bufferzeit.

Der Bandwith Tuner zeigt im Fenster RealFlash Bandwith Consumption anhand einer schwarzen Linie eine Kurve innerhalb unseres Movies an, welche die tatsächlich benötigte Datenübertragung für den Flash-Film darstellt. Wenn zum Beispiel an einer bestimmten Stelle im Flash-Film viele Symbole verwendet und animiert werden, müssen diese Daten kurzfristig zur Verfügung stehen. Damit es nicht zu stockenden Animationen kommt, können Sie die Bitrate des Flash-Films über einen Schieberegler an der linken Seite der Anzeige einstellen und gleichzeitig verfolgen, wie sich die Veränderung auf das Verhalten Ihres Flash-Films auswirkt. Auch hier gilt, nur der Test im Internet bringt Gewissheit über das Verhalten aller verwendeten Komponenten. Wer hier gründlich testet, geht sicher, dass seine Anwendungen auch funktionieren.

Sie sollten immer versuchen, die Bitrate von Flash-Filmen, die gleichzeitig mit Video- oder Audiodateien übertragen werden, möglichst niedrig zu halten.

Woher bekommen Sie den Bandwith Tuner?
Der Bandwith Tuner ist Bestandteil des Authoring Kit 8, das Sie kostenlos unter der folgenden Adresse von der RealNetwork-Website herunterladen können.
http://www.realnetworks.com/products/authkit/

Tipps zur Produktion

Flash CPU-Auslastung

Bei der Erstellung von Flash-Filmen für den Einsatz im RealPlayer sollten Sie darauf achten, keine zu komplexen Animationen zu verwenden. Bei zu komplizierten Animationen und Tween-Effekten kann die CPU-Belastung stark ansteigen, was dazu führen kann, dass nicht nur die Flash-Animation ins Stocken gerät, sondern auch eventuelle RealMedia-Daten nicht richtig wiedergegeben werden können. Je einfacher Ihre Flash-Filme aufgebaut sind, desto geringer ist die Gefahr, dass die Wiedergabe beeinträchtigt wird. In der Grundeinstellung verwenden Flash-Filme 12 Frames pro Sekunde. Wenn Sie Ihren Flash-Film mit anderen Dateien für die Wiedergabe im RealPlayer kombinieren möchten, sollten Sie die Framerate auf 9 bis 7 Frames pro Sekunde reduzieren.

Audio- und Flash-Movies

Versuchen Sie möglichst auf Sounddateien innerhalb Ihrer Flash-Filme zu verzichten. Abgesehen von kleinen Klicksounds, die von Buttons verwendet werden, rate ich von der Verwendung von Sounds hier ab, da diese die Dateigröße unseres Flash-Filmes nur ungünstig beeinflussen und das Streaming-Verhalten beeinträchtigen. Sie können mittels SMIL RealAudio-Dateien (.ra) dazu verwenden, Ihrem Flash-Film eine Hintergrundmusik oder eine Moderation zu verleihen. In diesem Fall wird die RealAudio-Datei gleichzeitig mit Ihrem Flash-Film gestreamt, was in jedem Fall die sicherste Technik dafür ist, beide Komponenten zeitgleich wiederzugeben.

Die richtige Auswahl des RealCodecs

RealVideo 8 ist der Standard-RealVideo-Codec. Der zur Verfügung stehende RealProducer bietet allerdings die Möglichkeit, zwischen dem RealVideo 8 Codec und zwei RealVideo G2 Codecs zu wählen. Die Wiedergabequalität des RealVideo 8 ist wesentlich besser als die beiden G2-Codecs, allerdings benötigt der RealVideo 8 Codec auch mehr Prozessorleistung und Zeit, um das Audio-/Videosignal zu dekodieren. Der RealPlayer 8 ist in der Lage, alle bisherigen Codecs korrekt darzustellen. Ältere RealPlayer können den RealVideo 8 Codec nicht wiedergeben. In diesem Fall wird der Anwender darauf hingewiesen, dass ein neueres Format verwendet wird, und erhält die Möglichkeit, seinen RealPlayer über die Update-Funktion zu aktualisieren.

Interaktive Kommandos

RealPlayer-Kommandos

Wie wir bereits erfahren haben, können innerhalb von Flash 3- und Flash 4-Filmen interaktive Kommandos verwendet werden, die den Ablauf des Films in sich steuern, hier kommen in der Regel `Play, Stop, Go to and Stop, Go To and` play sowie `GetURL` am häufigsten vor.

Neben diesen Kommandos können auch RealPlayer-Kommandos im Flash-Film verwendet werden. Hierdurch erhalten wir eine zusätzliche Möglichkeit, auf RealAudio- bzw. RealVideo-Dateien zuzugreifen und einzuwirken.

Es stehen derzeit die folgenden RealPlayer-Kommandos in Flash 3 und Flash 4 zur Verfügung:

- `Command:seek(time)`
  Dieses Kommando wird als GetURL-Kommando verwendet und ermöglicht einen genauen Zugriff auf die Timeline der aktuellen Präsentation. Beispiel: `command:seek(1.35.1)` springt an die angegebene Zeit in der Timeline der aktuellen Präsentation. Hierdurch können Szenen gezielt von Flash-Buttons aus aufgerufen werden. Auf die Timeline kann hiermit sogar mit einer Genauigkeit von einer Tausendstelsekunde (Millisekunde) zugegriffen werden.
- `command:seek(1.35.11.0)`
- `command:seek(tt,hh,mm,ss,xyz)`
  Erklärung: tt steht hier für Tage, hh für Stunden, mm für Minuten, ss für Sekunden, x steht für Zehntelsekunden, y für Hundertstelsekunden, und z steht für Tausendstelsekunden.

◀ **Abbildung 11**
Ein Beispiel für die Anwendung des seek-Kommandos

- `command:play()`
- `command:pause()`
- `command:stop()`

Die vorstehenden Kommandos stehen für den GetURL-Befehl in Flash und starten, pausen und stoppen den aktuellen Flash-Film.

Goto-Kommandos

Wenn Sie einem Flash-Film Interaktivität verleihen möchten, sollten Sie das mit dem Goto-Befehl tun. Sie sollten auf keinen Fall den Goto-Befehl dazu verwenden, innerhalb Ihres Flash-Films (im RealPlayer) von Szene zu Szene zu wechseln. In diesem Fall kann das RealPlayer-Kommando `command:seek` nicht mehr eingesetzt werden, da der Befehl Goto dessen Funktion außer Kraft setzt. Stellt der RealPlayer fest, dass im aktuellen Flash-Film ein Goto-Befehl verwendet wurde, wird der gesamte Flash-Film im Speicher zwischengespeichert und steht in voller Länge aus dem RAM des Computers zur Verfügung. Der Einsatz des Befehls Goto leitet das Zwischenspeichern ein, auf diese Weise wird es ermöglicht, den Sprung innerhalb des Flash-Films sofort ohne weiteres Nachladen zu ermöglichen.

Grundsätzlich sollten Sie versuchen, den Goto-Befehl nur einzusetzen, um interaktive Sprünge innerhalb Ihres Flash-Films zu realisieren. Vermeiden Sie es, mit dem Goto-Befehl innerhalb des Films von Szene zu Szene zu springen.

LoadMovie-Kommandos

Das LoadMovie-Kommando ermöglicht es, einen neuen Flash-Film in unsere Präsentation zu laden. Wir müssen also nicht diesen Befehl in unseren Flash-Film einbauen, sondern lösen diese Aufgabe durch den Einsatz von SMIL.

Hierdurch erhalten wir zusätzlich die Möglichkeit, mehrere Flash-Filme nacheinander zu laden und wiederzugeben. Zusätzlich können wir SMIL hier dazu verwenden, die Flash-Filme mit RealAudio-Dateien zu synchronisieren.

Das folgende Skript zeigt, wie zwei Flash-Filme nacheinander geladen und gleichzeitig mit RealVideo-Filmen wiedergegeben werden:

```
<smil>
   <body>
      <seq>
         <par>
            <animation src="rtsp://realserver.demo.com:55/
movie.swf"/>
            <audio src="rtsp://realserver.demo.com:55/
sound.rm"/>
         </par>
         <par>
            <animation src="rtsp://realserver.demo.com:55/
movi2.swf"/>
            <audio src="rtsp://realserver.demo.com:55/
sound2.rm"/>
         </par>
      </seq>
   </body>
</smil>
```

Sichere Transaktionen

In Flash 4 können Formulare erstellt werden, die zum Beispiel für Mail und Shopping-Lösungen Verwendung finden. Wird über ein Flash-Formular eine sichere Verbindung zu einem Server im Internet aufgebaut, so wird diese Verbindung an den Browser des Anwenders übergeben. Damit diese Übergabe funktioniert, muss der Anwender einen Browser auf seinem System installiert haben, der eine verschlüsselte Verbindung anneh-

men und darstellen kann. Der RealPlayer hat nicht die Möglichkeit, eine verschlüsselte Verbindung anzunehmen und darzustellen. Aus diesem Grund sind sichere Verbindungen mit dem RealPlayer derzeit nicht möglich.

Einen Flash-Film streamen

Hier noch einmal abschließend der Ablauf für die notwendigen Arbeiten, um einen Flash-Film im RealPlayer zu streamen.

1. **Exportieren Sie den Flash-Film (.swf)**
   Der RealServer kann nur Flash-Filme mit der Endung .swf (Shockwave-Flash-Format) streamen. Achten Sie darauf, dass Ihr Flash-Film möglichst keine Sounddateien enthält.
2. **Optimieren Sie den Flash-Film mit dem Bandwith-Tuner**
   Verwenden Sie den Bandwith-Tuner, um das Streaming-Verhalten Ihres Flash-Films optimal an Ihre Präsentation anzupassen. Sie haben die Möglichkeit, die genutzte Bitrate einzustellen, um hierdurch die Bufferzeit (das Zwischenspeichern) der benötigten Daten zu optimieren. Der Bandwith-Tuner ist Bestandteil des RealSystem Authoring Kit 8, das Sie unter dem folgenden Link herunterladen können:
   http://www.realnetworks.com/products/authkit/index.html
3. **Erstellen Sie eine Audiodatei**
   Wenn Sie Ihren Flash-Film mit einer Audiodatei synchronisieren möchten, sollten Sie in jedem Fall eine externe RealAudio-Datei verwenden und die Synchronisation mittels SMIL realisieren. Sie können die gewünschte Audiodatei zum Beispiel im WAV-Format mit Hilfe des RealProducers in das RealAudio-Fomat umwandeln. Nur Audiodateien im RealAudio-Format können im RealPlayer gestreamt werden. Achten Sie bei der Erstellung der RealAudio-Datei darauf, eine möglichst geringe Bittate auszuwählen, ohne die Qualität der Audiodatei zu beeinträchtigen.
4. **Fertigstellen der Präsentation**
   Damit nun alle Komponenten im RealPlayer wie gewünscht dargestellt werden, erstellen Sie eine SMIL-Datei, in der die Flash-Animation und die eben erstellte Audiodatei kombiniert werden. Testen Sie die SMIL-Datei erst lokal und anschließend auf Ihrem RealServer im Internet.

Nützliche Links zum Thema Flash und RealVideo:

RealNetworks Content Creation & Authoring Tools: http://www.realnetworks.com/devzone/library/creating/index.html

The SMIL Tutorial:
http://www.heli.org/products/smil/tutorial/

W3C SMIL Tutorial:
http://www.w3c.org/audioVideo/

Flash und QuickTime

Dieses Kapitel soll ein wenig über die Grenzen von Flash hinweg blicken. Hier wird Ihnen erklärt, wie Sie Bewegungsabläufe aus QuickTime zu Flash-Vektoren konvertieren. Bei Seiten wie egomedia.com oder powerflasher.de spielen sie eine zentrale Rolle.

Konvertierung von QuickTime-Video in Flash-Vektoren

▸ Danny Franzreb, www.franzreb.com

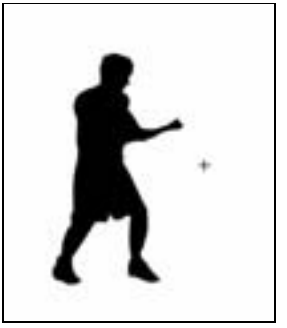
Wie Sie Bewegungs- und Formtweens erstellen, wissen Sie sicher schon, wie Sie einen QuickTime-Movie in Flash einbinden und steuern können, haben Sie vielleicht auch gelesen. Aber was machen Sie eigentlich, wenn Sie die natürlichen Bewegungen eines QuickTime-Movies und das platzsparende Flash-Format so kombinieren wollen, dass das Ergebnis auch noch über ein 56-kBit-Modem ohne vorzuladen streamt?

Arbeitsmaterialien – ein Überblick

Zuerst einmal möchte ich gerne klären, welche Komponenten zur Konvertierung von Movies in Flash-Vektoren nötig sind. Da wäre der QuickTime-Movie selbst, eine Aufnahme eines Bewegungsablaufs oder Ähnliches. Dann ein Programm, das es Ihnen ermöglicht, das Video ein wenig nachzuarbeiten und in Einzelbildern zu exportieren, hierzu wird beispielhaft Adobe Premiere eingesetzt. Die meisten anderen Video-Editing-Programme eignen sich genauso, da nur einfache Funktionen benötigt werden. Und zuletzt noch ein Grafikprogramm zum Nachbearbeiten dieser exportierten Bitmaps. In diesem Fall ist das Adobe Photoshop, weil es einfach als Industriestandard angesehen werden kann, hier gilt aber das Gleiche wie auch bei Premiere. Jedes andere Grafikprogramm, mit dem Pixelgrafiken bearbeitet werden können, kann auch verwendet werden, wie zum Beispiel Fireworks, Paintshop oder Photopaint.

Bitte öffnen Sie für die folgenden Erläuterungen die Beispieldateien auf der CD unter Zusammenspiel\Vektor.vid.

◀ **Abbildung 1**
Der QuickTime-Movie mit 375 KB



◀ **Abbildung 2**
Das Ziel: ein Flash-Film mit 8 KB

Umwandlung eines QuickTime Movies in einzelne Bitmaps

Nachdem Sie Premiere gestartet haben, wird Ihnen die Möglichkeit gegeben, einige Einstellungen für den zu bearbeitenden Film vorzunehmen. Hier sollten QuickTime und circa 30 fps ausgewählt werden. Logisch wäre eigentlich, hier schon 20 fps zu wählen, da Ihr Flash-Film später wohl auch in dieser Geschwindigkeit ablaufen wird. Dies schränkt Sie dann aber in der Auswahl der Bilder ein, die Sie nachbearbeiten können, da weniger Bilder exportiert werden.

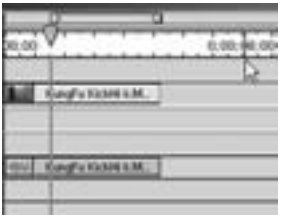
Bei den Einstellungen können auch die Farbtiefe und andere Werte des Films verändert werden. Diese Optionen sind für Ihre Zwecke aber nicht zwingend erforderlich, deswegen wird hier auch nicht näher darauf eingegangen.

Gute Voraussetzungen schaffen

Sollten Sie ein Projekt von Anfang an planen, empfiehlt es sich, die Bewegung vor weißem Hintergrund aufzuzeichnen und den Akteur dabei ganz in Schwarz einzukleiden, um so einen starken Kontrast zu erreichen. Das erleichtert die Nachbearbeitung ungemein, möglicherweise können Sie dann auf die Schritte in Photoshop ganz verzichten! In unserem Beispiel wird übrigens extra ein QuickTime Movie verwendet, der nicht vor ganz weißem Hintergrund aufgenommen wurde, um einige Probleme bei der Konvertierung zu verdeutlichen. Es ist natürlich ein Vorteil, wenn Sie von Anfang an besseres Material zur Verfügung haben. Dies ist aber eher selten der Fall.

Abbildung 3 ▸
Ziehen Sie das Video in Spur A

Abbildung 4 ▸
Die Zeitleiste, der Schieberegler und die Pfeile in Premiere



Danach öffnen Sie Ihren QuickTime-Movie über DATEI·ÖFFNEN bzw. (Strg+O).

Ziehen Sie Ihren Film jetzt auf die Zeitleiste, wobei Sie darauf achten sollten, dass sich der Film in der ersten Videospur befindet.

Jetzt können Sie über einen Klick in die Zeitleiste den Schieberegler aktivieren, mit dem Sie dann sehr genau den Anfang und das Ende der Bewegung in Ihrem QuickTime-Movie einstellen können. Setzen Sie den Regler zuerst auf den gewünschten Anfangspunkt im Film. Jetzt können Sie den linken Pfeil über den Schieberegler so verschieben, dass er genau über Ihrem Anfangspunkt einrastet. Setzen Sie den Schieberegler jetzt über den Endpunkt der Bewegung, und richten Sie den rechten Pfeil an ihm aus.

Sie haben nun den Teil des Filmes bestimmt, der später als Bitmap-Sequenz exportiert werden soll. Nun wollen wir noch ein paar Veränderungen an unserem Film vornehmen, die uns die Weiterverarbeitung in Photoshop erleichtern. Markieren Sie die erste Videospur, und wechseln Sie dann über Clips in das Menü FILTER bzw. (Strg+F). Da uns später nur noch Umrisse interessieren werden, wählen Sie hier bitte den Schwarzweißfilter und fügen diesen hinzu. Jetzt soll der Kontrast zwischen Vorder- und Hintergrund noch verstärkt werden. Das erreichen Sie über den

◀ **Abbildung 5**
Tonwertkorrektur in Premiere

Filter TONWERTKORREKTUR. Hier sollten Sie mit den drei Schiebereglern einfach ein wenig experimentieren, um die beste Einstellung für Ihr Projekt zu finden.

Damit haben Sie die Vorarbeiten in Premiere abgeschlossen und müssen den Film nur noch exportieren (DATEI·EXPORTIEREN oder Strg+M). Hier ist nur zu beachten, dass Sie danach unter EINSTELLUNGEN als Ausgabetyp WINDOWS BITMAP-SEQUENZ (BMP) oder TIFF-SEQUENZ (TIF) wählen und dass Sie nur den Arbeitsbereich exportieren.

Nun haben Sie zwei Möglichkeiten, Ihr Bildmaterial weiterzubearbeiten.

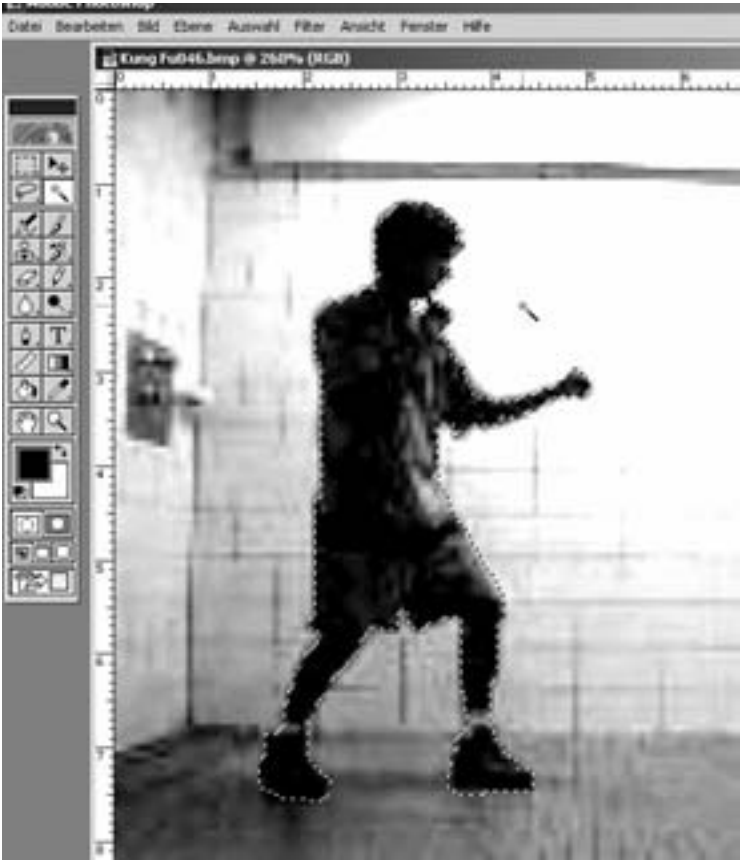
- Sollte Ihr QuickTime-Movie nicht unter optimalen Bedingungen aufgenommen worden sein, müssen verschiedene Elemente noch aus den einzelnen Bitmaps entfernt werden, bis später nur noch die von Ihnen gewünschte Bewegung als »Schatten« übrig bleibt.
- Wenn auf allen Bildern nur noch ein Schatten der Bewegung zu sehen ist, dann können Sie gleich bei »Finish in Flash« weiterlesen.

Nachbearbeitung im Grafikprogramm

In den meisten Fällen sind noch störende Elemente wie Teile des Hinter- oder Vordergrunds auf den Bitmaps zu sehen. Diese sind in Flash selbst nur mühsam zu entfernen, deswegen benutzen Sie hierzu besser ein Bildbearbeitungsprogramm mit komfortablen Auswahl und Cut-Funktionen. Das Vorgehen wird explizit am ersten Bild der Sequenz gezeigt. Sie müssen das gleiche Verfahren aber auf alle Bilder, die Sie später in Flash benutzen wollen, anwenden.

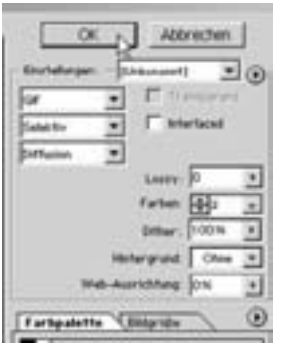
1. Öffnen Sie das Bild unter Photoshop (Datei · Öffnen), oder drücken Sie Strg+O.
2. Da in Premiere schon Tonwertkorrekturen vorgenommen worden sind, müsste Ihr Bild jetzt einen sehr hohen Kontrast aufweisen. Sollte dies nicht der Fall sein, können unter Bild · Einstellen · Tonwertkorrektur bzw. (Strg+L) noch Veränderungen vorgenommen werden. Der Ablauf ist der gleiche wie in Premiere.
3. Der hohe Kontrast des Bildes erleichtert Ihnen den nun folgenden Schritt erheblich. Wir wollen den Schatten der Bewegung nun von dem restlichen Bild befreien. Dazu müssen Sie mit dem Zauberstab auf den Schatten klicken. Sollte zu viel oder zu wenig ausgewählt werden, spielen Sie einfach ein bisschen mit der Toleranz des Zauberstabes (Zauberstab · Optionen · Toleranz). Ein hoher Wert (ab 30) sorgt hier dafür, dass mehr Pixeln mit unterschiedlichen Farbwerten ausgewählt werden. Ein niedriger Wert bewirkt das Gegenteil.
4. Falls die Auswahl immer noch nicht perfekt sein sollte, können Sie mit dem Lasso noch Teile hinzufügen. Halten Sie die ⇧-Taste gedrückt, und wählen Sie die Bereiche aus, die noch nicht durch den Zauberstab markiert wurden.
5. Nachdem der komplette Schatten von Ihnen ausgewählt wurde, füllen Sie ihn mit Schwarz (Bearbeiten · Fläche füllen). Wählen Sie unter Füllen mit die Option Vordergrundfarbe aus. Als Vordergrundfarbe muss vorher Schwarz aktiviert werden.
6. Nun kehren Sie die Auswahl um (Auswahl · Auswahl umkehren), oder verwenden Sie ⇧+Strg+I.
7. Jetzt löschen Sie den Hintergrund (Bearbeiten · löschen). Achten Sie hierbei darauf, dass als Hintergrundfarbe Weiss eingestellt ist.
8. Das Ergebnis können Sie nun speichern (Datei · Für Web speichern) (alternativ können Sie Alt+⇧+Strg+S drücken). Sie sollten das Format GIF wählen und unter Farben »2« eingeben. Diese Einstellungen garantieren Ihnen eine möglichst kleine Datei.

Damit sind die Vorarbeiten in Premiere und Photoshop abgeschlossen.



◂ **Abbildung 6**
Auswählen in Photoshop

▲ **Abbildung 7**
Füllen in Photoshop



▲ **Abbildung 8**
Die Einstellungen zum Speichern der GIFs

▲ Abbildung 9
Das GIF importiert in Flash

▲ Abbildung 10
Nachzeichnen

Finish in Flash

In Flash gibt es nun verschiedene Möglichkeiten, Ihre Sequenz noch weiter zu optimieren, die Sie je nach Geschmack einsetzen können.

Was Sie sicher tun müssen, ist, die einzelnen GIFs zu importieren (Datei · Importieren bzw. Strg + R). Wenn Sie das erste Bild importieren wollen, erkennt Flash normalerweise, dass es sich um eine Bildsequenz handelt, und bietet Ihnen die Möglichkeit, alle Bilder der Sequenz zu importieren. Flash weist jedem GIF ein Schlüsselbild zu.

Sie können die einzelnen GIFs noch in Vektoren umwandeln (Modifizieren · Bitmap nachzeichnen). Damit gehen dann oft Details des Schattens verloren, wenn Sie sich geschickt anstellen, nimmt der Shockwave-Movie aber weiter an Größe ab. Dazu sollten Sie für die Farbschwelle einen Wert um 100 Pixel wählen. Die Kurvenanpassung und Kantenschwelle darf auch nicht zu eng gewählt werden, sonst ist Ihre Vektorgrafik später größer als das GIF-Original!

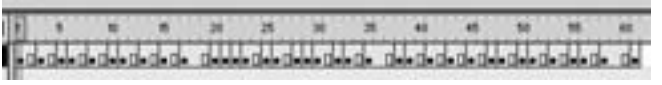
Um die Datei noch weiter zu verkleinern, können Sie die Kurven der Vektorgrafik noch optimieren (über Modifizieren · Kurven · Optimieren bwz. Strg + Alt + ⇧ + C).

Das beschriebene Verfahren muss nun für jedes Schlüsselbild wiederholt werden, und schon haben Sie ein QuickTime Movie vektorisiert.

Diese Methode zum Umwandeln von Bewegungsabläufen kann von Projekt zu Projekt leicht variieren. Wenn Sie sich an das hier vorgeführte Verfahren halten, sollten Sie aber früher oder später immer zum gewünschten Ergebnis gelangen. Viel Spaß beim Ausprobieren!

Automatisieren kann man den Vorgang des Vektorisierens von vielen Bildern übrigens mit dem Programm Adobe Streamline.

◀ **Abbildung 11**
Kurven optimieren



◀ **Abbildung 12**
Nach dem Löschen der Schlüsselbilder

▲ **Abbildung 13**
Schieben Sie die restlichen Bilder näher zusammen, um die Geschwindigkeit der Bewegung zu erhöhen.

Größe weiter optimieren

Oft passiert es, dass sich die einzelnen Bilder der Animation sehr ähnlich sind. Das heißt, Sie können leicht eins oder zwei Bilder zwischen jedem Schlüsselbild löschen, ohne dass der Betrachter dies bemerkt. Die Animation läuft weiter flüssig ab, und Sie haben die Größe der SWF-Datei nochmals um mehr als die Hälfte reduziert. Wenn Sie sich schon früh darüber klar werden, welche Bilder Sie nicht benötigen, können Sie eine Menge Zeit sparen, da Sie diese auch nicht in Photoshop und Flash bearbeiten müssen!

Wiedergabe von Videosequenzen in Flash

Wo ein Wille ist, ist auch ein Weg

Flash 5 ist kein Videoschnittprogramm und das Format SWF eignet sich nur bedingt für die Wiedergabe von Videosequenzen. Dennoch kommt man manchmal nicht um die Wiedergabe von Videosequenzen herum. Wir zeigen kurz, wie es geht.

▸ Dirk Schmeckthal, www.flash2be.com

Unser primäres Ziel ist es, den Videofilm zunächst einmal in Einzelbilder zu zerlegen, die dann in Flash weiterverarbeitet werden können. Hierzu gibt es zwei verschiedene Möglichkeiten.

Verschiedene Ansprüche brauchen verschiedene Lösungen. Im Detail kann man das wegen der Komplexität leider kaum besprechen, aber einige Methoden möchten wir vorstellen.

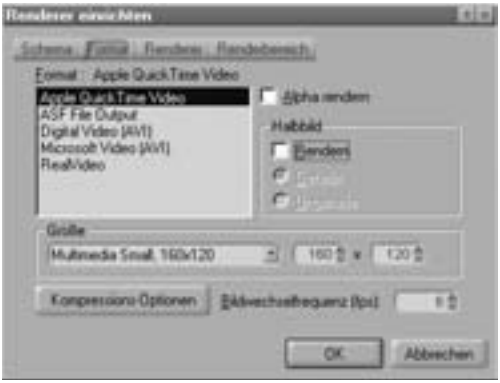
- In Methode Eins wird der Movie-Explorer von »Aist« verwendet, weil die Basic Version mehrsprachig und kostenlos ist. Das Programm unterstützt diverse Im- und Exportformate (gängige Streaming-Formate wie RealMedia, QuickTime, MS ASF). Methode Eins macht aus einem 37 Sekunden dauernden Film (avi 240 x 180, 5.34 MB) ein SWF in der Größe von 698 KB.
- Die zweite Methode (JPEG) geht einen sehr direkten Weg. Es wird ausgiebig stapelverarbeitet (gebatched). Dafür verwenden wir die RAD-Tools. Sie macht aus einem 37 Sekunden dauernden Film (avi 240 x 180, 5.34 MB) ein SWF in der Größe von 494 KB.

Wichtig: Für bessere Bildqualität sind Videostreaming-Formate geeigneter.

Mithilfe des Movie-Explorers

Voraussetzungen:

- Viel Zeit und noch mehr RAM
- Apple QuickTime 4
- Videoeditiersoftware / Aist Movie-Explorer, Adobe Premiere, Apple QuickTime Pro o.Ä.



▲ **Abbildung 1**
Menüpunkt Renderer einrichten

▲ **Abbildung 2**
Compression Settings

Als erstes müssen Sie eine Filmdatei (*.avi, *.mpeg o.ä.) in einem der oben genannten Programme als QuickTime (*.mov) stark komprimiert exportieren.

Die Vorgehensweise mit dem Movie-Explorer ist die Folgende: (Die Komprimierungseinstellungen sind in Adobe Premiere und QuickTime Pro sowie in anderen Programmen, die QuickTime unterstützen, identisch.)

Nach dem Start des Programms werden Sie aufgefordert, die Animation einzurichten. In unserem Fall ist es unerheblich, was Sie hier wählen, da wir nur am späteren Export interessiert sind.

Geben Sie unter FORMAT·APPLE QUICKTIME VIDEO ein. Als BILDWECHSELFREQUENZ wählen Sie 6 fps. Die Wahl der Größe bleibt Ihnen überlassen, empfehlenswert ist aufgrund der Dateigröße aber eher ein kleineres Format.

In den Kompressions-Optionen sollten Sie unter COMPRESSOR·SORENSON VIDEO einstellen. Die FRAMES PER SECOND werden auf 6 gestellt.

Die QUALITY steht hier, wie Sie in Abbildung 2 sehen, auf LOW. Dies hat den Vorteil, dass wir kleine .fla-Dateien erhalten, diese haben aber auch eine schlechte Qualität. Besser ist es, hier einen Wert wie BEST auszuwählen. Die eigentliche Komprimierung werden wir ja am Ende erst in Flash durchführen.

Als RENDERBEREICH geben Sie unter RENDERER EINRICHTEN die Einstellung STATISCH·ALLES an. Diese Einstellung speichern Sie als Schema, um beim Export auf diese Voreinstellung zurückgreifen zu können.

Abbildung 3 ▸
Sie speichern ein Schema

Im Fenster Browser/Desktop klicken Sie auf den Ordner, in dem sich Ihr Film befindet. Den Film können Sie nun in die Zeitleiste auf eine Videospur ziehen. Das Dialogfenster Quellen kann mit OK bestätigt werden.

Beim Exportieren bzw. Rendern (Strg+R) muss jetzt nur noch das abgespeicherte Schema gewählt werden.

Mithilfe der RAD-Tools

Die zweite Methode mittels JPEG geht einen sehr direkten Weg. Es wird ausgiebig stapelverarbeitet (gebatched). Wir verwenden hierfür die RAD-Tools. Weitere Informationen zu den RAD-Videotools finden Sie unter www.radgametools.com.

Als erstes zerlegt man den Quellfilm in eine BMP-Sequenz. Die RAD-Videotools bieten dafür eine schnelle Hilfe:

Man markiert den Quellfilm und klickt auf den Button Convert file. Klicken Sie anschließend auf Browse und wählen Sie im darauffolgenden Menü den Zielordner aus. Output type wird auf BMP gestellt und die Bildrate auf 6 fps verringert. Dies geschieht unter dem Punkt Adjust (adds/remove frames) to.

Durch einen Klick auf Convert starten Sie den Vorgang. RAD zerlegt in Windeseile den Film in Einzelbilder im BMP-Format.

Es ist nicht sinnvoll, diese Bilder noch in einem Grafikbearbeitungsprogramm (z. B. Fireworks) vorbereiten, da Flash BMPs sehr gut als JPG komprimiert.

◀ Abbildung 4
Das Rendern

Der Flash-Import

Auf welchem Weg man nun auch immer die Einzelbilder erhalten hat, sie müssen nun in Flash importiert werden.

In Flash legen Sie ein neues Dokument an. Setzen Sie die BPS-Rate in den FILMEIGENSCHAFTEN ebenfalls auf 6 Bilder pro Sekunde und stellen Sie die gewünschte Bühnengröße ein. Einen Trick, mit dessen Hilfe man auch mit höheren Bildraten gute Ergebnisse erzielt, ohne mehr Bilder importieren zu müssen, verraten wir im How-to-Teil unter »SSX«, siehe Seite 315.

Nun importiert man das mit dem Movie-Explorer erzeugte QuickTime-Movie und fügt mit der Taste F5 die Bilder hinzu, bis der ganze Film sichtbar ist. Diesen Film können wir nun als BMP-Sequenz exportieren.

Importieren Sie das erste Bild nun auf eine neue Ebene. Flash fragt Sie automatisch, ob die gesamte Sequenz hinzugefügt werden soll, was Sie bestätigen. Ist das geschehen, kann die QuickTime-Ebene gelöscht werden.

Sollten Sie diesen Film jetzt exportieren wollen, müssen Sie beim Export nur noch die JPG-Einstellungen Ihren Wünschen anpassen.

Flash und JavaScript

Flash und JavaScript arbeiten vorzüglich zusammen. So kann JavaScript z. B. Ereignisse von Flashfilmen empfangen und über Methoden des Flash-Players den Flashfilm steuern.

Kommunikation zwischen Flashfiles in verschiedenen Frames

▸ Thomas Weller, www.weller-net.de

Dieses Tutorial soll die Kommunikation zwischen Flashfiles in verschiedenen Frames der HTML-Seite und die Kommunikation zwischen Flash und HTML-Objekten erläutern. Die Kommunikation erfolgt per JavaScript.

Kommunikation mit der HTML-Seite per JavaScript

JavaScript ist eine auf HTML-Seiten eingesetzte Scriptsprache, die in der Lage ist, mit dem Browser zu kommunizieren und Objekte auf der HTML-Seite anzusprechen. In diesem Tutorial sprechen wir JavaScript mit dem Flash-Aktion `GetURL` an.

Als Beispiel ein Funktionsaufruf in Flash 4,

```
on (release) {
   getURL ("JavaScript:meinefunction();");
}
```

und im JavaScript Teil der HTML-Seite stände dann die Funktion

```
meinefunction (){
   machwas;
}
```

Für einen einfachen Funktionsaufruf im selben Frame, in dem sich das Flashfile befindet, könnte man auch die FSCommand-Aktion in Flash benutzen. Dies würde folgendermaßen aufgerufen werden

```
on (release) {
   fscommand ("Befehl", "Argument");
}
```

und im JavaScript Teil der HTML-Seite stünde dann diese Funktion:

```
function Flashfilename_DoFSCommand(command, args){
   machwas;
}
```

Warum nun mit GetURL? Mit FSCommand ist man in der Lage, auf den JavaScript-Teil des Frames zuzugreifen, in dem sich das Flashfile befindet. Mit GetURL kann man direkt auf Funktionen und Objekte in anderen Frames zugreifen und wäre sogar in der Lage, ganz ohne JavaScript-Teil im HTML-File auszukommen.

Was wird dazu gebraucht?

Um die Kommunikation zwischen JavaScript und Flash zu ermöglichen, muss man im HTML-Teil der Seite einiges beachten. Da mit Objekten in den anderen Frames kommuniziert werden soll, sollten diese Frames eindeutig mit Namen definiert werden. Ebenso müssen die Flashfiles eindeutige Namen erhalten. Manchmal muss man einen großen Aufwand betreiben, um in den verschiedenen Browsern ein einheitliches Ergebnis zu erzielen. Dies zwingt uns auch in diesem Fall, verschiedene Wege zu gehen.

- Der Microsoft Internet Explorer nutzt für das Einbinden von Flash-Files den <objekt>-Tag. In diesem muss zusätzlich der Parameter id = "Flashfilename" angegeben werden.
- Der Netscape Navigator nutzt den <embed>-Tag, darin müssen zusätzlich die Parameter name = "Flashfilename" und swliveconnect = true angegeben werden. Da swliveconnect = true die Java Engine des Browsers lädt und dies einige Zeit in Anspruch nimmt, sollte man diesen Parameter nur nutzen, wenn Flashfiles mit JavaScript kommunizieren.

Ebenso wie beim Einbinden der Files verwenden die Browser beim Ansprechen eine andere Syntax. Aus diesem Grund erstellen wir eine Variable `ie`, welche den Browser identifiziert, diese Variable erhält den Wert `true`, wenn es sich um den Microsoft Internet Explorer handelt. Diese Variable sollte am besten gleich im Head-Bereich der Seite definiert werden und könnte natürlich auch für andere browserspezifische Funktionen im JavaScript-Teil genutzt werden.

```
var ie = navigator.appName.indexOf("Microsoft") != -1;
```

Des Weiteren nutzen wir zum Ansprechen der Flashfiles eine Variable `m`, welche den Wert `parent.Framename.Flashfilename` enthält, wenn es um den Microsoft Internet Explorer geht und `parent.Framename.document.embeds[0]`, wenn es sich nicht um den Microsoft Internet Explorer handelt.

```
var m = ie ? parent.Framename.Flashfilename : parent.Framename.
document.embeds[0];
```

Nun folgt die Syntaxerklärung. Für die Flash-User, die sich bisher nur wenig mit JavaScript beschäftigt haben, dies ab der Version 5 von Flash aber beabsichtigen, folgen an dieser Stelle einige Erklärungen zu der bis jetzt benutzten Syntax.

Die eben genutzten Variablendefinitionen sind eine JavaScript-Kurzform, manchmal wirkt diese Form aber sehr unübersichtlich, aus diesem Grund soll hier die »Langform« angegeben werden. `navigator` ist ein Objekt des vom User benutzten Browsers, mit `indexOf` wird die Startposition des Strings `"Microsoft"` gesucht. Wird er nicht gefunden, enthält `indexOf` den Wert –1 und würde somit der Variablen `ie` den Wert `false` zuweisen.

```
var ie = navigator.appName.indexOf("Microsoft") != -1;
```

Dies ist gleichbedeutend mit:

```
if (navigator.appName.indexOf("Microsoft") != -1){
   var ie = true;
}else{
   var ie = false;
}
```

Mit `parent.Framename` wählen wir das gewünschte Frame der HTML-Seite und sind damit in der Lage, auf definierte Objekte dieses Frames zuzugreifen. Der Microsoft Internet Explorer spricht das Flashfile direkt mit Namen an. Der Netscape Navigator tut dies in der Reihenfolge, in der die Objekte eingebunden wurden.

```
var m = ie ? parent.Framename.Flashfilename :
parent.Framename.document.embeds[0];
```

Dies ist gleichbedeutend mit:

```
if (ie){  //
hier wird abgefragt ob die Variable ie true ist
   var m = parent.Framename.Flashfilename;
}else{
   var m = parent.Framename.document.embeds[0];
}
```

Das Beispiel – Ausführung von Aktionen im anderen Flashfile

Ausführung von Aktionen:

- in einem anderen Flashfile: `document.Flashfile.Aktion()`
- in einem anderen Frame `parent.Framename.Flashfile.Aktion().`

Für die verschiedenen Browser rufen wir nun unsere Variable `m` mit nachfolgendem `Aktion` auf: `m.Aktion()`

Dies sieht dann im Flashfile folgendermaßen aus:

```
on (release) {
   getURL ("JavaScript:var m = ie ? parent.Framename.Flashfilename :
parent.Framename.document.embeds[0]; m.Aktion()");
}
```

Aus dem HTML-Teil der Seite würde man diese Aktion so aufrufen:

```
<a href= "JavaScript:var m = ie ? parent.Framename.Flashfilename :
parent.Framename.document.embeds[0]; m.Aktion()">Machwas</a>
```

Die im Beispiel verwendeten Aktionen

Syntax	Flashfile / Button	Beschreibung
`m.GotoFrame(4)`	filmx.swf / anordnen	geht zu Bild 4 in (framey)
`m.TGotoFrame("_flash0",14)`	filmx.swf / loadMovie	geht zu Bild 14 auf Ebene 0 (in framey)
`m.TGotoLabel("_flash0","objekte")`	filmx.swf / Objekte	geht zu Label `"objekte"` auf Ebene 0 (in framey)
`m.TGotoFrame("_flash0", m.TCurrentFrame("_flash0")+1)`	filmy.swf / vor Button	geht (in framex) ein Bild vor
`m.TGotoFrame("_flash0", m.TCurrentFrame("_flash0")-1)`	filmy.swf / zurück Button	geht (in framex) ein Bild zurück
`m.LoadMovie(1, 'images/ movie1.swf');`	filmy.swf / load Movie	lädt `"movie1.swf"` auf Ebene 1 (in framez)
`m.TPlay('_flash1')`	filmy.swf / play	startet `"movie1.swf"` auf Ebene 1 (in framez)
`m.TStopPlay('_flash1')`	filmy.swf / stop	stoppt `"movie1.swf"` auf Ebene 1 (in framez)
`m.LoadMovie(1, '')`	filmy.swf / unload	löscht `"movie1.swf"` auf Ebene 1 (in framez)

Funktionen mit Variablenübergabe bzw. Auslesen der Variable

Um Variablen an ein anderes Frame zu übergeben bzw. aus einem anderen Flashfile auszulesen, geht man genauso vor.

Im Beispiel wird unter dem Menüpunkt ANORDNEN, Button 1 und Button 2, jeweils die JavaScript-Funktion aufgerufen und eine Variable übergeben.

```
on (release) {
   getURL ("javascript:parent.framez.wechsel(1);");
}
```

Unter den Buttons 3 und 4 wird die Variable definiert und die JavaScript-Funktion aufgerufen.

```
on (release) {
   oben = 3;
   getURL ("javascript:parent.framez.wechsel();");
}
```

Die Variablendefinition und die JavaScript-Funktion im `framez`.

```
var ie = navigator.appName.indexOf("Microsoft") != -1;
var m = ie ? parent.framey.filmy : parent.framey.
document.embeds[0];

function wechsel(xn){
if (xn) {
   uebergabe = xn;
}else{
   uebergabe = m.GetVariable("oben");
}
   var_label = "Redraw";
   document.window_reihenfolge.SetVariable("/:NewTop",uebergabe);
   document.window_reihenfolge.TGotoLabel("/control/",var_label);
}
```

Wenn man zum Funktionsaufruf eine Variable übergibt, wird der erste Teil der IF-Schleife ausgeführt. Im ELSE-Teil wird die Variable mit der Aktion `m.GetVariable("oben")` ausgelesen.

Ebenso kann man Variablen im Flashfile manipulieren. Dies geschieht im unteren Teil der Funktion.

```
document.window_reihenfolge.SetVariable("/:NewTop",uebergabe);
```

In diesem Fall lautet die Syntax `document.Flashfile.Aktion`, da das Flashfile im selben Frame angesprochen wird. `/:NewTop` ist der Variablenname, wobei `/:` angibt, dass es sich um eine Variable handelt. In `uebergabe` steht der neue Wert der Variablen.

Danach wird zum Bild-Label Redraw in der Filmsequenz mit Instanznamen `control` gesprungen.

```
document.window_reihenfolge.TGotoLabel("/control/",var_label);
```

Die übergebene Variable wird dann in diesem Label verarbeitet. Wie man JavaScript-Variablen aus Flash ändert, wird im folgenden Abschnitt mit aufgeführt.

Objektmanipulation

Man kann – wie schon gesagt – auch ohne JavaScript im HTML-File auskommen. Im Beispiel ist folgende GetURL-Anweisung unter dem Menüpunkt OBJEKTE·BUTTON 1 dargestellt. Dabei wird die JavaScript-Variable ausgegeben und das Objekt bild1.src manipuliert.

```
on (release) {
   GetURL ("JavaScript:parent.framez.ausgeben = true;parent.
framez.document.bild1.src = 'images/1.gif'; target=framez"),
}
```

Dies funktioniert nur richtig, wenn man nochmals das Frame angibt, worin das Objekt geändert werden soll. Nachfolgend sehen Sie das Ganze nochmals mit einem Funktionsaufruf, bei dem die Angabe des Framenamens nicht notwendig ist.

```
on (release) {
   getURL ("JavaScript:parent.framez.ausgeben = false; parent.
framez.tauschebild('1');");
}
```

In die GetURL-Anweisung kann man aber auch komplexere JavaScript-Anweisungen einbinden. Möchte man eventuell seine JavaScript-Anweisungen nicht für jeden gleich sichtbar machen, so wäre dies eine Möglichkeit.

```
on (release) {
   getURL ("JavaScript:zaehler = (zahlda)? zaehler+1 : 1;zahlda = true;
if (ie) {
   parent.framez.document.all.Layer1.innerText = 'Klick '+zaehler;}
else if (ns) {
   with (parent.framez.document.Layer1.document){
      open(); write('Klick '+zaehler);
      close();
   }
}
target=framez");
}
```

Im Beispiel wird die JavaScript-Variable `zaehler` erstellt, hochgezählt und im Layer1 ausgegeben. Somit sollte der Kommunikation zwischen Flashfiles, Objekten und dem Browser nichts mehr im Wege stehen.

Weiterführende Informationen

Für den Macromedia Dreamweaver gibt es diverse Erweiterungen bezüglich der Steuerung von Flashfiles auf der Homepage von Macromedia. Ein weiteres Beispiel ist auf der Flash 4 CD im Verzeichnis \Goodies\Macromedia\JavaScriptDemo zu finden. Darin findet die Kommunikation aus dem Flashfile mit dem Aktion `FSCommand` statt.

Flashfilm bildschirmfüllend – Randlos glücklich

◂ Daniel Schulten, www.netzkern.com

Endlich ist der Flashfilm, an dem man so lange gearbeitet hat, fertig, und man fragt sich, wie man ihn nun geschicktesten ins Internet bringt. Es reicht sicherlich nicht aus, die Datei einfach auf den Server zu spielen, man muss vielmehr, wie bei Grafiken auch, einer HTML-Datei sagen, wo der Flashfilm stehen soll.

- Ein Flashfilm kann als einzelnes Objekt auf einer Seite stehen, genauso wie eine gewöhnliche Grafik auch. Den dazu notwendigen Code kann ich mit Veröffentlichen einfach erzeugen lassen und per Cut & Paste an die passende Stelle einfügen (oder z. B. in Dreamweaver wie eine Grafik einfügen).
- Ich kann meinen Flashfilm aber auch **fensterfüllend** anzeigen lassen, d.h. ein Browserfenster vollkommen damit ausfüllen, indem ich ein Frameset erstelle und einen Frame mit meinem Flashfilm fülle. Wer dies per Veröffentlichen schon einmal versucht hat, wird feststellen, dass er einen circa 5 Pixel breiten Rand rund um den Flashfilm erhält, außerdem wird ein eigentlich überflüssiger Scrollbalken auftauchen.

Wie man nun rand- und scrollbalkenlos glücklich wird, wie ich meinen Flashfilm in einem neuen Fenster öffne, in einem Teil des Bildschirms oder im Vollbild, lesen Sie im Folgenden.

Die Möglichkeiten

Es existieren bereits zwei relativ einfache Wege, ein gutes Ergebnis zu erzielen.

- Zum einen gibt's den berühmten Body-Tag:
  ```
  <BODY topmargin="0" leftmargin="0" marginwidth="0"
  marginheight="0">
  ```
- Außerdem steht das Stylesheet zur Verfügung. Achtung, das gehört natürlich in den Head!
  ```
  <style type="text/css"> body { margin-left:0px; }
  { margin-top:0px; } </style>
  ```

Manchmal genügen diese beiden Möglichkeiten aber nicht, dann bietet sich noch eine weitere Möglichkeit an – Die Frame-Lösung.

Die Frame-Lösung

Nachdem der Flashfilm fertiggestellt ist, sollten Sie diesen mittels Veröffentlichen automatisch in eine HTML-Datei einbinden lassen. Ich bezeichne von nun an alles in englischer Sprache, weil man es dann im Quelltext besser wiedererkennt.

In den Publish Settings (Veröffentlichen) sollte Folgendes eingestellt sein:

In der Registerkarte Format sollte die Option Flash und HTML eingestellt sein. Wenn die Namen nicht modifiziert werden sollen, merken Sie sich an dieser Stelle den Namen der HTML-Datei! In unserem Beispiel ist dies flash.htm.

Die Registerkarte Flash braucht nicht modifiziert zu werden. In der Registerkarte HTML stellen Sie für Dimensions·Percent Width/Height jeweils auf 100.

Alle weiteren Einträge können beibehalten werden, gegebenenfalls kann man die Einstellungen unter Scale ändern. Dabei bedeutet

- Default (Show all)
  Mein Flashfilm wird soweit groß oder klein skaliert, bis eine Dimension meines Filmes den Fensterrand erreicht, also zum Beispiel die horizontale Ausrichtung maximal ist. In der anderen Richtung wird man über den Rand der Bühnengröße hinausblicken. Dies gilt es bei der Positionierung von Elementen außerhalb der Bühne zu beachten! Die Seitenverhältnisse bleiben erhalten.

- No Border
  Vergleichbar mit Default, allerdings können Teile des Films abgeschnitten werden. Die Seitenverhältnisse bleiben ebenfalls erhalten.
- Exact fit
  Hierbei wird der Film in beide Richtungen maximal eingepasst, mit der Folge, dass der Film verzerrt werden könnte. Bereiche außerhalb der Bühne sind nicht sichtbar.

Meiner Meinung nach fährt man mit der Default-Einstellung am besten. Jetzt kann man mit Publish die SWF- und HTML-Datei erzeugen.

In einem Teil des Fensters

Habe ich eine Navigation mit Flash erstellt, die in einem Teil der Seite permanent zu sehen sein soll, kann ich wie auch unter HTML mit einem Frameset arbeiten. Dabei ist es dann egal, ob in den anderen Frames normale HTML-Seiten stehen oder ebenfalls Flashfilme abgespielt werden sollen. Wichtig ist, dieses Frameset korrekt zu tunen.

```
<frameset cols="15%,*" frameborder="0" border=0 framespacing="0">
   <frame name="flash" src="flash.htm" marginwidth="0"
marginheight="0" scrolling="Auto" frameborder="0" noresize
framespacing="0">
   <frame name="inhalt" src="inhalt.htm" marginwidth="0"
marginheight="0" scrolling="Auto" frameborder="0" noresize
framespacing="0">
</frameset>
```

Hierbei ist die Datei »flash.htm« die von Flash erzeugte HTML-Datei, in der mein Flashfilm läuft, die Datei »inhalt.htm« kann eine normale HTML-Seite sein oder auch einen Flashfilm beinhalten.

Verlinkungen aus dem Flashfilm in die Inhaltsseite kann ich per `getUrl` einfügen und muss dabei lediglich als Target »inhalt« angeben.

Wichtig beim Anpassen auf die eigenen Bedürfnisse ist, dass man bei den Frames, die Flash enthalten, unter `scrolling` unbedingt »No« einstellen sollte, da ein Flashfilm nicht gescrollt werden muss. `Marginwidth` und `Marginheight` sowie `Frameborder` schalten sowohl unter dem Netscape Navigator/Communicator als auch unter dem Internet Explorer die Ränder rund um den Flashfilm aus, `Frameborder 0` setzt beide Frames direkt nebeneinander.

Fensterfüllend

Möchte ich nur einen Flashfilm im gesamten Fenster öffnen, muss ich das obere Frameset lediglich geringfügig abändern.

```
<frameset cols="100%,*" frameborder="0" border=0 framespacing="0">
   <frame name="haupt" src="flash.htm" marginwidth="0"
marginheight="0" scrolling="Auto" frameborder="0"
noresize framespacing="0">
   <frame>
</frameset>
```

So erreiche ich, dass meine Datei flash.htm über das gesamte Fenster angezeigt wird. Da der andere Frame ohnehin nicht sichtbar ist, brauche ich hier keinen Namen und auch keine Quelle angeben.

Doch was mache ich, wenn mein Flashfilm gar nicht auf die volle Fenstergröße skaliert werden soll? Dann gibt es wiederum zwei Möglichkeiten, die beide auf den schon erwähnten Methoden aufbauen, diese aber nicht ersetzen!

Im gleichen Fenster

Wenn ich kein neues Browserfenster öffnen will, kann ich die oben genannten Framesets in ein weiteres Frameset »einbetten«. Dabei übernimmt das übergestülpte Frameset die Rolle eines Rahmens.

```
<frameset rows="*,560,*" frameborder="0" border=0
framespacing="0">
   <frame>
   <frameset cols="*,760,*" frameborder="0" border=0
framespacing="0">
      <frame>
      <frame name="inhaltt" src="frameset.htm" marginwidth="0"
marginheight="0" scrolling="Auto" frameborder="0" noresize
framespacing="0">
      <frame>
   </frameset >
   <frame>
</frameset>
```

Hierbei ist die Datei »frameset.htm« eines der beiden oben erwähnten Framesets, sie funktioniert mit beiden. Im ersten Fall habe ich so eine Navigationsleiste und den Inhalt in der festen Größe, im zweiten Fall füllt der Flashfilm komplett die Mitte des Fensters aus.

Zu beachten ist, dass die äußeren Frames, der Rahmen, weiß sind. Habe ich einen Flashfilm mit einer anderen Hintergrundfarbe, sollte ich hier anstelle von `"<frame>"` explizit Quelldateien angeben, die einen passenden Hintergrund haben.

In einem neuen Fenster

Diese Variante findet man sehr oft bei Flashseiten, sie ist mit einem kleinen JavaScript sehr einfach zu verwirklichen.

```
<script language="JavaScript">
<!--
function macheFenster()
   {
   var newWindow = window.open('flash.htm','flash',
'status=yes,menubar=yes,scrollbars=yes,width=760,height=560')
   }
// -->
</script>
```

Diese Funktion muss man in die Seite einbauen, aus der man per Mausklick ein neues Fenster öffnen will. Dabei kann der Link `javascript:macheFenster()` auch in einem Flashfilm stehen und per `GetUrl` aufgerufen werden. Mehr dazu im übernächsten Kapitel. Ebenso kann man zum Beispiel die Größe oder die zu öffnende URL erst über den Funktionsaufruf per Variable eingeben.

Und nun viel Spaß und Erfolg beim Ausprobieren!

Position einer Filmsequenz an Flash übergeben

◂
boblgum
www.mysterion.de

Wenn Sie Ihren Flashfilm bildschirmfüllend ablaufen lassen, wie auf den vorherigen Seiten erläutert, so kann es zu merkwürdigen Effekten kommen. So haben wir mit Dreamweaver 3 eine Seite erstellt, die aus drei Frames bestand (oben, mitte, unten). Im mittleren Frame sollte der

Flash-Film zu sehen sein. Es fiel aber auf, dass unser Flash-Film auf eine sehr seltsame Art skaliert wurde.

Obwohl die Einstellungen auf 100 % in beide Richtungen standen, wurden die Grafiken nicht wie üblich verzerrt. Vielmehr wurde nur der Hintergrund des Films auf 100 % der Framegröße skaliert. Das Ganze sah natürlich nicht mehr so aus, wie wir es uns wünschten. Deswegen soll Folgendes versucht werden:

- Die sichtbare Größe des Browserfensters ermitteln.
- Die ermittelten Werte an Flash übergeben.
- Anhand dieser Werte die »Bauteile« des Films positionieren

Obwohl der Netscape Navigator meiner Meinung nach bessere Möglichkeiten bietet, die clientseitige Konfiguration des Browsers zu ermitteln, funktioniert die Positionierung nicht so überzeugend wie im Internet Explorer. Ich konnte zwar die Variablen an Flash übergeben, aber derselbe Rechenweg ergab im Netscape Navigator seltsame Ergebnisse.

Zu Testzwecken habe ich den Netscape Navigator-Teil des JavaScripts im HTML-Code der Seite gelassen (ist nicht aktiviert). Was ich auf meinem System darauf feststellen konnte:

- Die Übergabe der Variablen im Netscape Navigator klappte nur mittels eines `FSCommand`,
- `onResize` hat im Netscape Navigator überhaupt nicht funktioniert :((deswegen `FSCommand`).

Der HTML-Code der Seite entspricht den Standardeinstellungen bei der Veröffentlichung aus Flash (»Flash only«).

Hier folgt der Auszug aus dem JavaScript speziell für den Internet Explorer:

```
function IEsize()
   {
   var InternetExplorer = navigator.appName.indexOf("Microsoft")
!= -1;
   var path = InternetExplorer ? movieIE : document.embeds[0];
   if(navigator.appName == "Microsoft Internet Explorer")
      {
      path.SetVariable('Xsize', document.body.clientWidth);
      }
   }
```

Die Variable wird mit `path.SetVariable('Xsize', document.body.clientWidth);` an Flash übergeben und, wenn diese im Film noch nicht vorhanden ist, erzeugt. Der Befehl setzt sich zusammen aus:

- `path`: Name des Films (Kann man auch durch den ID-Wert aus dem `OBJECT` ersetzen)
- `SetVariable`: Die Anweisung an Flash
- `Xsize`: Name der Variablen (in einfachen Anführungszeichen!)
- `document.body.clientWidth`: Wert der Variablen
- Aktiviert wird das Ganze durch die Anweisungen im BODY `onLoad="IESize()" und onResize="IEsize()"`
- `movieIE` entspricht dem ID-Wert im `OBJECT`-Teil des Seiten-Codes

Nun kommen wir zum Flash-Teil. Wie man sehen kann, steht der Filmaufbau in der Mitte des Frames. Wenn man aber »außerhalb« des Films die rechte Maustaste anklickt, erscheint das Flash-Kontextmenü. Daraus folgt: nur der Hintergrund des Films wurde skaliert.

Zu beachten ist aber auch, daß die Koordinaten dieselben geblieben sind. Das heißt, dass x=0 NICHT am linken Rand des Browserfensters liegt! Dies ist wichtig, um die Position richtig bestimmen zu können.

Der Flashfilm (Breite=x) liegt genau in der Mitte des Frames. `Xsize` sei die sichtbare Fenstergröße. Daraus folgt:

```
x(Rechter Rand) = x + (Xsize - x)/2
x(Linker Rand) = 0 - (Xsize - x)/2
```

Um die Breite des Films zu ermitteln, setzen wir einen MovieClip genau in die Mitte des Films (Strg+K bzw. An Seite ausrichten). Dem MovieClip geben wir den Instanznamen `x`.

Positionierung eines MovieClips am rechten Rand des Fensters

ActionScript-Beispiel:

```
1: x = 2* _root.xMC._x;
2: if (x < Xsize) {
3:    _root.bild._x = x + (Xsize - x) / 2 - _root.bild._width / 2;
4: }else {
5:    _root.bild._x = x - _root.bild._width / 2;
6: }
```

Zeile 1. Es wird eine neue Variable `x` deklariert, die den Wert `"2* _root.xMC._x"` hat. Speziell in unserem Fall bedeutet das folgendes: Es wird die Position (x-Wert) der Instanz `xMC` ermittelt (`xMC` ist der Movie-Clip, den wir in der Mitte des Films positioniert haben). Um die Breite des Films zu errechnen, verdoppeln wir den ermittelten x-Wert.

Zeile 2. Eine IF-Anweisung wird eingeleitet. Diese Anweisung wird in diesem Fall nur dann ausgeführt, wenn die Variable `x` den Wert kleiner als `Xsize` (die Fensterbreite) hat.

Zeile 3. Hier wird die neue Position der Instanz `bild` bestimmt. Wir ermitteln dazu die Breite der zu positionierenden Instanz `bild` und halbieren diesen, um die Mitte zu finden. Die Mitte entspricht der x-Position der Instanz. Um `bild` genau am Rand des Fensters zu platzieren, errechnen wir die Strecke von x=0 bis zum Rand (`x+ (Xsize - x) / 2`) und ziehen die Hälfte der Breite der Instanz ab.

Zeile 4. Wenn das Fenster aber »kleiner« als der Film ist:

Zeile 5. Die Position von `bild` wird durch das Abziehen der halbierten Breite von der Fenstergröße ermittelt. Warum? Weil Flash die Größe des Films beim Verkleinern nicht ändert! Wenn der Film also 600 px breit war, dann bleibt er für Flash immer 600 px breit. Auch, wenn es anders aussieht :)

Was noch fehlt, ist eine Schleife, durch die unser ActionScript immer wieder durchlaufen wird. Dies kann man zum Beispiel durch einen `goto`-Befehl erreichen.

Zusätzliche Browserfenster öffnen

▸ Michael Bundscherer, www.typolis.de

Aufpoppende Browserfenster können ganz schön nervend sein! Besonders häufig auf Seiten mit erotischen Inhalten springen einem die neuen (Werbe-)Fenster nur so entgegen – habe ich mir sagen lassen.

Aber ein neues Fenster zu öffnen, kann (besonders auf Flash-Seiten) auch sinnvoll sein. Häufig wird hier auf besondere Produkte oder Aktionen hingewiesen, die – irgendwo auf der Seite platziert – nur untergehen oder das Layout zerstören würden. Längerer Text ist auf Flash-Seiten nur schwer lesbar. Auch hier bietet sich ein neues Fenster mit einem HTML-Text an. Und zu guter Letzt kann es ja auch sein, dass eine Flash-Seite mit vorgegebener Größe in einem neuen Fenster ohne Rand geöffnet werden soll.

Um aus Flash heraus ein neues Browser-Fenster zu öffnen, fügt man entweder einem Schlüsselbild (Keyframe) oder einer Schaltfläche (But-

ton) die `getURL`-Aktion hinzu. Wenn dann das Schlüsselbild erreicht oder die Schaltfläche geklickt wird, öffnet sich die angegebene Seite in einem neuen Browser-Fenster. Allerdings ist das ein ganz normales Fenster; die Größe ist vom Entwickler nicht beeinflussbar und die Navigationsleisten wirken manchmal störend.

Mit JavaScript kann man das Aussehen eines neuen Fensters ziemlich gut beeinflussen, verwenden wir das doch auch für unsere Zwecke.

Hier geht's um die Einstellung

Wir haben uns also entschieden, das neue Fenster mit Hilfe von JavaScript zu öffnen. Hier kann man viele Parameter einstellen, die das Aussehen des Fensters bestimmen:

- `toolbar` = die Navigationsleiste (yes/no oder 1/0)
- `menubar` = die Menüleiste (yes/no oder 1/0)
- `location` = die Adressleiste (yes/no oder 1/0)
- `status` = Statuszeile (yes/no oder 1/0)
- `resizable` = vergrößern des Fensters möglich (yes/no oder 1/0)
- `scrollbars` = Scrollbalken (yes/no oder 1/0)
- `fullscreen` = Inhalt wird fensterfüllend angezeigt (yes/no oder 1/0)
- `width` = Weite (Wert in Pixel)
- `height` = Höhe (Wert in Pixel)
- `left` = Abstand zum linken Monitorrand (Wert in Pixel)
- `top` = Abstand zum oberen Monitorrand (Wert in Pixel)
- `name` = Name des Fensters, um es mit anderen JavaScripten ansprechen zu können

Sie sollten sich aber ganz genau überlegen, was Sie alles weglassen wollen. Es gibt viele Leute, die es sehr störend finden, wenn zum Beispiel die Statuszeile ausgeblendet ist. Wie will man denn dann erkennen, ob die Seite schon fertig geladen ist? Und wenn die Menüleiste fehlt, kann man weder die Seite drucken, noch sie in die Bookmarks legen.

Weitere Einstellungen

Es gibt noch mehr Einstellungsmöglichkeiten. Viele werden aber wieder mal nicht von allen Browsern unterstützt. Welche das sind, können Sie auf http://www.webreference.com/js/column7/attributes.html erfahren.

So funktioniert es: keine Angst vor JavaScript

Im Prinzip funktioniert das so, wie es am Anfang beschrieben wurde: mit der `getURL`-Aktion. Nur rufen wir hier jetzt keine einfache HTML-Seite auf, sondern ein JavaScript. Die Funktion, die dann ausgeführt werden soll, wird dann noch in der HTML-Seite definiert. Man beachte: wenn jetzt JavaScript verwendet wird, muss dies auch beim Benutzer aktiviert sein, damit das Fenster aufgeht!

Das kommt in die HTML-Seite:

```
<HEAD>
<TITLE>Hier steht der Seiten-Titel</TITLE>

<!-- Flash-Fenster-Popper v1.0 - www.flashworker.de -->
<SCRIPT LANGUAGE="JavaScript"> var version = 1.0;</SCRIPT>
<SCRIPT LANGUAGE="JavaScript1.1"> version = 1.1;</SCRIPT>

<SCRIPT LANGUAGE="JavaScript">
var newwin;
function flashworker_popup(url,name,eigenschaften)
   {
   newwin = window.open(url,name,eigenschaften);
   if(version > 1.0)
      {
      setTimeout('newwin.focus();',200);
      }
   }
</SCRIPT>
</HEAD>
<BODY>
Hier kommt der Inhalt der Seite hin (also der HTML-Text oder das SWF).
</BODY>
</HTML>
```

Erklärung: Zunächst einmal wird abgefragt, ob JavaScript Version 1.0 oder 1.1 aktiv ist. Dann kommt die Funktion, die zum Öffnen des Fensters zuständig ist. Wenn JavaScript 1.1 vorhanden ist, wird das neu geöffnete Fenster nach 0,2 Sekunden fokussiert, um sicher zu gehen, dass das neue Fenster nicht von anderen Browserfenstern verdeckt wird.

Das kommt (zum Beispiel) in Flash auf eine getURL-Aktion:

```
javascript:flashworker_popup('url.htm','Fenster-Name',
'toolbar=0,menubar=0,location=0,status=1,scrollbars=0,resizable=1,
width=525,height=380')
```

Erklärung: Hier wird also nicht eine HTML-Seite aufgerufen, sondern unsere JavaScript-Funktion, die in der HTML-Seite eingebettet ist. In der Klammer stehen dann die ganzen Zusätze wie die URL, der JavaScript-Name des Fensters und dessen Eigenschaften. Wenn man übrigens irgendwelche Eigenschaften nicht definiert (wenn ich hier zum Beispiel `scrollbars=0` weglasse), dann werden die Standardeinstellungen des Browsers verwendet.

Das war alles?

Ja, war es eigentlich schon. Bedenken sollte man vielleicht noch, dass der Netscape Navigator und der Internet Explorer die definierte Breite und Höhe des Pop-up-Fensters (width und height) wieder unterschiedlich interpretieren. Der Internet Explorer meint, Breite ist alles innerhalb des Fensters, Netscape hingegen nimmt als Breite die Breite des ganzen Fensters. Netscape macht also die eigentliche Breite (und Höhe) der HTML-Seite immer einige Pixel kleiner. Um einen Wert für die Breite und Höhe des Fensterinhalts anzugeben, kann man ab dem Netscape Navigator 4 statt dessen die Parameter `innerwidth` und `innerheight` verwenden. Natürlich wird das dann von älteren Netscape-Browsern oder gar vom Internet Explorer nicht verstanden, was also auch keine Hilfe ist. Im Zweifelsfall sollte der Inhalt also immer in den Netscape-Browser passen!

Wenn es mit bestimmten Browser noch Probleme gibt, habe ich weiter unten noch einen Lösungsvorschlag.

Und wieder weg damit

Nehmen wir an, wir haben gerade ein Fenster geöffnet, wie kann man es wieder schließen? Das geht wieder ziemlich simpel mit einem JavaScript, der ebenfalls auf eine getURL-Aktion gesetzt wird.

Das kommt in den Flashfilm als Aktion auf einen Button bzw. auf ein Frame (`getURL`):

Je nach Struktur der HTML-Seiten:

- `javascript:self.close()`
- `javascript:parent.close()`
- `javascript:top.close()`
- `javascript:window.close()`

Noch eine Lösung

Man kann es natürlich noch viel umständlicher machen. Und zwar dann, wenn das Pop-up-Fenster auch auf allen Browsern funktionieren soll. Die eben beschriebene Lösung funktioniert nämlich nicht mit dem Internet Explorer 3.0 auf dem PC und IE 4.5 auf dem Mac. Aber es gibt eine Lösung, wie es auch auf diesen Browsern klappt (diese Lösung wird allerdings wiederum vom IE 3 auf dem Mac nicht unterstützt):

Zuerst erstellen Sie ein Frameset, bei dem das erste Frame (mit dem SWF-Film) die Höhe 100% besitzt und das zweite die Höhe *. Das zweite Frame ist für den Benutzer also nicht sichtbar. Wir geben ihm – damit wir es später ansprechen können – noch einen Namen, zum Beispiel »Pop-up«.

Beim Aufruf des Framesets befindet sich im zweiten Frame einfach eine »leere« HTML-Datei. Jetzt braucht man für jedes Pop-up-Fenster noch eine extra HTML-Datei, die dieses Fenster mit einem `OnLoad`-Befehl im Body-Tag aufruft.

Und das war es dann schon! Das Ganze könnte dann beispielsweise so ausschauen:

Das Frameset:

```
<HTML>
<HEAD>
<TITLE>Meine tolle Flash-Seite</TITLE>

<FRAMESET rows="100%,*" border="0">
   <FRAME src="filmurl.htm" name="Film" frameborder="0" noresize>
   <FRAME src="PopUpLeer.htm" name="PopUp" frameborder="0"
noresize>
</FRAMESET>

</HEAD>
</HTML>
```

Die HTML-Datei, die im zweiten Frame aufgerufen wird:

```
<HEAD>
<TITLE>PopUp-Fenster-1</TITLE>

<!-- HTML-Fenster-Popper v1.0 - www.flashworker.de -->
<SCRIPT LANGUAGE="JavaScript"> var version = 1.0;</SCRIPT>
<SCRIPT LANGUAGE="JavaScript1.1"> version = 1.1;</SCRIPT>

<SCRIPT LANGUAGE="JavaScript">
var newwin;
function flashworker_popup(url,name,eigenschaften)
   {
   newwin = window.open(url,name,eigenschaften);
   if(version > 1.0)
      {
      setTimeout('newwin.focus();',200);
      }
   }
</SCRIPT>
</HEAD>

<BODY onLoad="flashworker_popup('url.htm','Fenster-Name',
'toolbar=0,menubar=0,location=0,status=1,scrollbars=0,resizable=1,
width=525,height=380')">
</BODY>
</HTML>
```

Der Aufruf aus der Flash-Datei:

```
getURL ("PopUp-Fenster-1.htm", "PopUp");
```

Weitere Infos

Weitere Infos zu Pop-up-Fenstern aus Flash gibt es zum Beispiel unter www.moock.org/webdesign/flash (englisch).
Ein nützliches Script, welches den oben gezeigten Code automatisch erzeugt, findet man unter http://www.flashworker.de/gox.htm?tutorial/42/003.html

Abbildung 1 ▶
Dieses Browserfenster haben wir erstellt

PopUp-Fenster aus Flash
Datei Bearbeiten Ansicht Favoriten Extras ?
So schaut das Browser-Fenster mit Deinen Einstellungen aus.
[Fenster schließen]
Internet

Flash 5 JavaScript-Integration

Neue Funktionen für Flash 5-Formulare

Das JavaScript Integration Kit für Flash 5 bietet die Möglichkeit, die Funktionalität von Flash 5-Formularen zu erweitern. In diesem Kapitel erfahren Sie, wie Sie Macromedia Dreamweaver dazu einsetzen können, Ihren Flash-Formularen mehr Power zu verleihen.

▸ Detlef Randerath
www.detlef.de

Formulare sind seit jeher sehr beliebte Kommunikationsschnittstellen im Internet. Für einen Flash-Autoren passen HTML-Formulare allerdings nicht unbedingt gut in das Gesamtbild einer in Flash entwickelten Website. Seit der Veröffentlichung von Flash 4 können Formulare auch direkt in Flash entwickelt werden.

In der Regel werden diese Formulare für den Versand von E-Mail-Nachrichten verwendet. HTML-Formulare wurden seit Erscheinen von Flash 4 immer seltenere Gäste in Flash-Websites. Ein Problem bestand leider darin, logische Überprüfungen in diesen Flash-Formularen durchzuführen. Bisher musste die dazu benötigte Logik mittels ActionScript in Flash erstellt werden, was nicht unbedingt jedem Anwender von Flash leicht von der Hand geht.

Das JavaScript Integration Kit für Macromedia Flash 5 ermöglicht den Einsatz von vorgefertigten JavaScript-Prüfroutinen direkt in der HTML-Seite, in die unser Flash-Formular integriert ist. JavaScript wird hier eingesetzt, um die benötigten logischen Prüfungen wie zum Beispiel die Überprüfung der Eingabe einer Postleitzahl oder einer E-Mail-Adresse clientseitig, also im Browser des Anwenders durchzuführen. Es sind keine weiteren Datenbankabfragen auf externen Servern im Internet mehr nötig, um logische Prüfungen durchzuführen.

Dieses Kapitel unterteilt sich im Folgenden in drei Teilbereiche für das JavaScript Integration Kit für Flash 5. Im ersten Bereich widmen wir uns den Flash Player-Kontrollen (Flash Player Controls) und deren Funktionen. Im zweiten Bereich lernen wir den Umgang für die erweiterten For-

mularprüfungen kennen, und der abschließende dritte Bereich erklärt uns den Umgang mit den neuen Browser-Skripten für Flash 5.

Das benötigen wir

Wie bereits erwähnt, setzen wir in diesem Kapitel neben Flash 5 auch Macromedia Dreamweaver 3 ein. Für den Fall, dass Sie noch keine Version von Dreamweaver 3 besitzen, auf der CD-ROM zum Buch ist eine Trial-Version für Sie abgelegt, mit der Sie die folgenden Übungen ausführen können. Es ist übrigens wichtig, dass wir Dreamweaver 3 einsetzen, da das JavaScript Integration Kit für Flash 5 nur zu dieser Version von Dreamweaver kompatibel ist.

Neben Flash 5 und Dreamweaver 3 benötigen wir eine zusätzliche Komponente für Dreamweaver, den »Extension Manager 1.1«, mit dessen Hilfe das JavaScript Integration Kit als Erweiterung in Dreamweaver installiert werden kann. Auch hier ist es wichtig, dass Version 1.1 verwendet wird. Als letzte Komponente benötigen wir natürlich das JavaScript Integration Kit für Flash 5.

◀ siehe CD-ROM

Ich möchte die Gelegenheit nutzen und Ihnen die Vorteile des Macromedia Exchange-Dienstes im Internet nahe zu legen. Sie finden den Exchange-Bereich auf der deutschen Macromedia Website unter der folgenden Internetadresse: http://www.macromedia.com/de/ im Menüpunkt Club. Wer sich hier als Mitglied registrieren lässt, erhält eine Macromedia-ID, die als Zugangsberechtigung für den Exchange-Bereicht gilt. Hier finden Sie ständig aktuell die Neuheiten zu Macromedia Dreamweaver und können auf ein riesiges Archiv von Erweiterungen für den Dreamweaver zugreifen und diese herunterladen. Besonders von Vorteil ist natürlich, dass dieser Service von Macromedia kostenlos für alle Dreamweaver-Anwender angeboten wird.

Zum Zeitpunkt der Drucklegung dieses Buches stand das JavaScript Integration Kit für Flash 5 nur in der englischen Version zur Verfügung. Nach Angaben von Macromedia Europe ist derzeit nicht geplant, dass Macromedia eine deutsche Version des JavaScript Integration Kits herausbringt.

Systemanforderungen für die Macromedia Flash Player-Kontrolle:

Die Flash Player-Kontrolle basiert auf JavaScript und arbeitet mit Netscape und Microsoft Internet Explorer 3.0 und neueren Versionen, allerdings nicht mit Microsoft Internet Explorer 4.5 und 5.0 auf dem Macintosh. Die Verhalten Set Flash by List, Load Flash Movie und das Verhalten

Fast Forward funktionieren nur im Internet Explorer. Bei den eingesetzten Browsern muss JavaScript eingeschaltet und eine aktuelle Flash Player-Version installiert sein.

Für den Einsatz der erweiterten Formularprüfungen und Browser-Skripten für Flash benötigt man einen Netscape- oder Microsoft-Browser mit der Versionsnummer 3.0 oder eine neuere Version. JavaScript muss für die korrekte Funktion der Skripten eingeschaltet sowie ein aktueller Flash Player installiert sein.

Vorbereitungen

Installieren Sie Dreamweaver 3 und den Extensions Manager. Kopieren Sie nun aus dem Ordner JavaScript Integration/Software die Datei MX100630_mmJIK.mxp in den Ordner Macromedia/Dreamweaver3/Downloaded Extensions und starten Dreamweaver anschließend.

Im Menüpunkt Kommandos finden Sie nun den Eintrag Manage Extensions.

Wählen Sie diesen Menüpunkt aus. Das Fenster, das sich nun öffnet, repräsentiert den Extensions Manager. Wählen Sie nun mit dem Tastaturkürzel Strg+I die Funktion Install new Extensions aus.

Der Extension Manager öffnet daraufhin ein Auswahlfenster, das den Inhalt des eben verwendeten Ordners Downloaded Extensions zeigt, in dem sich jetzt das JavaScript Integration Kit befindet. Wählen Sie nun die Datei MX100630_mmJIK.mxp aus. Folgen Sie den Anweisungen der Installation, anschließend können Sie das JavaScript Integration Kit – wie in Abbildung 1 gezeigt – im Auswahlfenster auswählen.

Im unteren der beiden Fenster des Extension Managers werden Detailinformationen über die bereits installierten Erweiterungen in Dreamweaver angezeigt. In unserem Fall werden Informationen zu den im JavaScript Integration Kit enthaltenen JavaScripts angezeigt. Hier finden Sie Informationen zu den Erweiterungen, den Versionsnummern, dem Typ der Erweiterungen und den Namen des Autors.

Erst wenn Sie das JavaScript Integration Kit hier angezeigt bekommen, können Sie auch mit den Verhalten (Behaviors) in Dreamweaver arbeiten. Sollte die Erweiterung hier nicht angezeigt werden, wiederholen Sie die Installation.

Extension Manager
File Help
Dreamweaver
Installed Extensions
Version
Type
Author
Button-O-Matic 1.2.0 Command Brendan Dawes
Counter Wizard - Add a Counter 1.0.3 Object MyComputer.com
Country Codes 1.0.0 Object Rabi Sunder Raj
GuestBook Wizard - Add a Guestbook 1.0.1 Object MyComputer.com
JavaScript Integration Kit for Macromedia Flash 5 1.0.0 Suite Macromedia
Phone.com WML 1.1 DTD 1.0.0 Browser Profile Danny Angus
Print Link 1.0.0 Object Goussard Laurent
SMIL 1.0 DTD Browser Profile 1.0.0 Browser Profile Danny Angus
Sound 2.1.0 Object Rabi Sunder Raj
Swap Flash 1.2 Behavior Massimo Foti
W3C HTML 4.01 Strict DTD Browser Profile 1.0.2 Browser Profile Danny Angus
WAP Forum WML 1.1 DTD 1.0.0 Browser Profile Danny Angus
Zero Page Borders 1.0 Command Andrew Wooldridge
+ Macromedia Flash Player Controls
Provides VCR-like controls to Macromedia Flash animations. Using Dreamweaver Behaviors easily add play, stop, fast-forward, rewind, and zoom controls to any HTML object on the page. It can load new Flash movies from a drop-down list.
+ Advanced Form Validations
Use Dreamweaver to add 18 pre-written client-side form validations to your Flash Forms. Build Forms in Flash and then use Dreamweaver to add advanced form validation.
+ Browser Scripts for Flash
Easily reference a library of common JavaScript in Flash ActionScript. Browser Scripts for Flash provide encapsulated JavaScript for working with Cookies, interacting with HTML frames, opening new windows and controlling images.
+ Flash Deployment Kit

◀ **Abbildung 1**
Der Extension Manager informiert ausführlich über installierte Erweiterungen

Was sind Verhalten? (Behaviors)

Verhalten beschreiben Handlungen, die von einer Aktion des Anwenders ausgehen. Wenn zum Beispiel ein Anwender den Mauszeiger über einen Textlink positioniert, wird dieser in einer anderen Farbe dargestellt. Die auslösende Handlung war in diesem Fall das Positionieren des Mauszeigers über dem Textlink, was wiederum in der Aktion des Hervorhebens des Textlinks in einer anderen Farbe endet. Aktionen werden auf der Basis von vorgefertigten JavaScripts ausgeführt. Der Dreamweaver erlaubt das Zuweisen von Handlungen und Aktionen in verschiedenen Konstellationen.

Mit den Verhalten, die im JavaScript Integration Kit für Flash 5 enthalten sind, können Handlungen und Aktionen zugewiesen werden, die den Ablauf unseres Flash-Films kontrollieren und verändern können. Die Macromedia Flash Player-Kontrollen ermöglichen es zum Beispiel, eine Grafik als Stop-Button zu deklarieren. Das JavaScript »Stop Flash« wird der Grafik im Dreamweaver zugewiesen und stoppt dann, wenn die Grafik angeklickt wird, unseren Flash-Film. In diesem Zusammenhang sollten

Abbildung 2 ▶
Der Behaviors-(Verhalten-) Inspektor mit ausgewählter Handlung (Event)

Sie beachten, dass die Handlungen, die wir zum Auslösen der Aktionen benötigen, von Browser zu Browser unterschiedlich sein können. Der Behaviors-Inspektor (Verhalten-Inspektor) ermöglicht eine genaue Auswahl im Bereich Handlungen (Events), um die gewünschte Handlung auswählen zu können.

Ihr Event ist nicht verfügbar?
Sollte das gewünschte Event nicht im Drop-down-Menü zur Verfügung stehen, so können Sie wie hier gezeigt das `onMouseOver`-Event auswählen und dann manuell im HTML-Fenster ändern und gegen den gewünschten z. B. `onClick` austauschen.

Wichtig ist, dass Sie hier die richtige Auswahl für die Browser treffen, für die Sie Ihre Webseiten erstellen möchten. Beachten Sie bitte in diesem Zusammenhang die Einschränkungen der Flash Player-Kontrollen (Flash Player Controls).

Flash Player-Kontrollen in der Übersicht

Die Verhalten (Behaviors), die im JavaScript Integration Kit für Flash 5 enthalten sind, ermöglichen es Ihnen auf einfache Art und Weise, mittels Flash 5 und dem Macromedia Dreamweaver interaktive Flash-Anwendungen für Ihre Website zu entwickeln. Besonders nützlich sind die Flash Player-Kontrollen bei der Verwendung von bereits fertig gestellten Flash-Filmen, die auf zusätzliche Art neue Verwendung in Ihrer Website finden sollen, indem sie mittels der Flash Player-Kontrollen neue Funktionalitäten erlangen.

Wir wollen die folgenden Beispiele dazu nutzen, um ein Steuerpult für unseren Flash-Film in unserer HTML-Seite mittels einiger kleiner Grafiken zu erstellen.

◄ Abbildung 3
Wählen Sie hier die ID für den Flash-Film aus, den Sie vorspulen möchten

Die dazu benötigten Grafiken und die vorgefertigte HTML-Datei finden Sie im Ordner JavaScript Integration/Übung. Kopieren Sie diesen Ordner auf Ihre Festplatte, damit Sie die Dateien lokal modifizieren und speichern können. Öffnen Sie die Datei Übung.htm in Dreamweaver, und führen Sie die nachfolgend erklärten Abläufe selbst aus, um ein Gefühl für die Funktionen der Flash Player-Kontrolle zu bekommen.

Sie können natürlich auch während der Übungen mit verschiedenen Komponenten experimentieren, um das Verhalten der verschiedenen Objekte im Zusammenhang mit der Flash Player-Kontrolle kennen zu lernen.

Schnelles Vorspulen (Fast Forward Flash)

Verwenden Sie das Verhalten Fast Forward Flash, um aus einem Objekt in Ihrer Website einen Schnellvorlauf-Button für Ihren Flash-Film zu erstellen.

Gehen Sie bei diesem und jedem folgenden Verhalten wie folgt vor:

Klicken Sie auf den Button Flash-Film-einfügen (Insert Flash) in Dreamweaver, und wählen Sie den Flash-Film Cityanimation.swf aus dem Ordner Übungen aus, um diesen in Ihre HTML-Seite einzubinden. Vergeben Sie für den Flash-Film einen Namen und eine ID. Achten Sie darauf, dass die Namen in beiden Feldern gleich sind, damit diese als Referenz für den eben eingefügten Flash-Film verwendet werden können.

Wählen Sie nun die Grafik »FastForward« aus. Klicken Sie anschließend im Verhalten-Inspektor (Behaviors Inspector) das Plussymbol an und wählen das Verhalten Fast Forward Flash aus dem Untermenü Macromedia Flash Player Controls aus.

Wählen Sie nun im Fenster Fast Forward Flash den Film aus, auf den sich das Objekt beziehen soll, und stellen Sie über die Eingabe-/Auswahlmöglichkeiten die Parameter ein, die beim Zugriff auf den Flash-Film verwendet werden sollen.

Geben Sie im Eingabefeld Fast Forward die Zahl 10 ein und wählen im Dropdown-Menü Frames aus. Nun wird der Flash-Film bei jedem Klicken auf das Objekt um jeweils zehn Frames vorgespult.

Die Voreinstellung für die initiierende Handlung für dieses Verhalten ist je nach ausgewähltem Zielbrowser unterschiedlich, diese Einstellung finden Sie im Behaviors-Inspektor. Sie können die Einstellung allerdings im Behaviors-Inspektor gegen andere Handlungen (Events) austauschen, je nachdem, für welche Browser die Seite entwickelt wurde.

Nachdem Sie das Verhalten durch Anklicken aktiviert haben, wird der Flash-Film nicht weiterlaufen, sondern an der Stelle, die Sie aufgerufen haben, stoppen. Damit der Flash-Film nach dem Aufruf weiterläuft, müssen Sie zusätzlich nach dem Fast Forward Flash-Verhalten noch das Verhalten Play Flash hinzufügen.

Damit es nicht zu unerwarteten Effekten bei der Wiedergabe kommt, sollten Sie darauf achten, zuerst das aufrufende Verhalten und dann im Anschluss das Verhalten Play Flash zuzuweisen.

Gehe zu Flash Frame (Go to Flash Frame)

Verwenden Sie das Verhalten Go to Flash Frame, um mit einem ausgewählten Objekt wie z. B. einer Grafik auf Ihrer Webseite, einen speziellen Frame in Ihrem Flash-Film aufzurufen.

Gehe zu Flash Frame laut Cookie (Go to Flash Frame Based on Cookie)

Verwenden Sie dieses Verhalten, wenn Sie die Information, die in einem bereits auf dem Computer des Anwenders gespeicherten Cookie liegt, verwenden möchten, um den Anwender zu einem bestimmten Frame in Ihrem Flash-Film zu führen.

Ein Beispiel wäre, dass der Anwender einen bestimmten Trainingsfilm gesehen hat und Sie ihn nun zur gezeigten Animation befragen möchten.

Weisen Sie der Grafik mit der Beschriftung »Cookie« ein neues Verhalten zu.

◄ Abbildung 4
So sollten die beschriebenen Einstellungen anschließend aussehen

Lade Flash-Film (Load Flash Movie)

Diese Flash Player-Kontrolle ermöglicht es Ihnen, den in Ihrer HTML-Seite aktuell angezeigten Flash-Film durch einen anderen Film zu ersetzen.

Weisen Sie der Grafik »Load Flash« ein neues Verhalten zu.

Stellen Sie im LOAD FLASH MOVIE-Dialogfenster nun folgende Einstellungen ein. Der Parameter REPLACE MOVIE sollte bereits unseren Flash-Film anzeigen, sollte dies nicht der Fall sein, wählen Sie im Drop-down-Menü die ID für unseren Film »City« aus. Der folgende Parameter WITH MOVIE verweist auf den zu ladenden Flash-Film, der die Stelle unseres ursprünglichen Flash-Films einnehmen soll.

Benutzen Sie den Button BROWSE..., um nach dem gewünschten Film zu suchen. Im Eingabefeld LEVEL lassen wir den voreingestellten Wert 0 stehen. Hierdurch wird der ursprünglich dargestellte Film aus dem Speicher entfernt und der neue Film an seiner Stelle in der HTML-Seite dargestellt. Als Letztes stellen wir noch über den Parameter PLAY ein, ob der neugeladene Flash-Film automatisch gestartet werden soll oder nicht.

Achten Sie bitte bei der Verwendung dieses Verhaltens darauf, dass derzeit nur der Microsoft Internet Explorer in der Lage ist, dieses Verhalten richtig auszuführen. Sie sollten diese Tatsache bei der Planung Ihrer Flash-Anwendungen berücksichtigen. Anwender anderer Browser, wie zum Beispiel Netscape oder Opera Browser, sind möglicherweise durch diese Tatsache nicht in der Lage, Ihre Website richtig zu nutzen.

Film starten (Play Flash)

Verwenden Sie dieses Verhalten, um ein Objekt in Ihrer HTML-Seite in einen PLAY-Button zu verwandeln, der die Wiedergabe Ihres Flash-Films starten kann.

Weisen Sie der Grafik »Play« ein neues Verhalten zu.

Im Gegensatz zu manch anderen Verhalten, die wir hier kennen gelernt haben, funktioniert das Verhalten Play Flash, ohne dass weitere Verhalten dazu notwendig sind. Wenn Sie also einem Objekt in Ihrer HTML-Seite, wie zum Beispiel einer Grafik, das Verhalten Play Flash zugewiesen haben, startet der betroffene Flash-Film, sobald das Verhalten aktiviert wird.

Bei der Verwendung der übrigen Verhalten ist bis auf wenige Ausnahmen die zusätzliche Verwendung des Verhaltens Play Flash ratsam, damit diese Verhalten ihre Funktion auch auf den entsprechenden Flash-Film anwenden können.

Film zurückspulen (Rewind Flash)

Verwenden Sie dieses Verhalten, um den Flash-Film in Ihrer HTML-Seite zurückzuspulen. Dieses Verhalten stoppt die Wiedergabe des Films und kehrt zum Frame Nummer eins zurück.

Denken Sie auch hier daran, noch das Verhalten Play Flash hinzuzufügen.

Film stoppen (Stop Flash)

Verwenden Sie dieses Verhalten, um aus einem Objekt, zum Beispiel einer Grafik, in Ihrer HTML-Seite einen Stop-Button für Ihren Flash-Film zu erstellen. Wird dieses Verhalten ausgeführt, wird die Wiedergabe des Flash-Films gestoppt.

Weisen Sie der Grafik »Stop« ein neues Verhalten zu.

Im Gegensatz zu manchen anderen Verhalten tritt die Wirkung dieses Verhaltens sofort nach dem Aufruf in Kraft. Es ist also nicht notwendig, weitere Verhalten zuzuweisen.

◄ Abbildung 5
Wählen Sie die ID für den Flash-Film aus, den Sie zoomen möchten

Zoom Flash (Zoom Flash)

Verwenden Sie dieses Verhalten, um aus einem Objekt wie zum Beispiel einer Grafik in Ihrer HTML-Seite einen Zoom-Button zu erstellen. Dieses Verhalten lässt Sie bei jeder Verwendung des Buttons weiter in einen ausgewählten Flash-Film hinein zoomen.

Weisen Sie der Grafik »Zoom« ein neues Verhalten zu.

Damit Sie dieses Verhalten gleichermaßen für das Vergrößern als auch für das Verkleinern der Darstellung Ihres Flash-Films einsetzen können, ist es wichtig, die Zahlensysteme für dieses Verhalten zu beachten.

Alle Zahlenwerte kleiner/gleich (<=) 100 vergrößern den Flash-Film und zoomen in den Film hinein. Alle Zahlen, die größer als der Wert 100 sind, verkleinern den Flash-Film und zoomen aus. So können Sie dieses Verhalten für Ihren Button VERGRÖSSERN und für Ihren VERKLEINERN verwenden.

Sollte der Flash-Film schon wiedergegeben werden, während Sie dieses Verhalten verwenden, so wird die Wiedergabe in der Regel durch die Zoomfunktion nicht beeinträchtigt.

Flash aus Liste auswählen (Set Flash by List)

Dieses Verhalten bietet Ihnen die Möglichkeit, innerhalb einer HTML-Seite verschiedene Flash-Filme aus einer Auswahlliste auszuwählen und darzustellen. Dies bietet Ihnen zum Beispiel die Möglichkeit, Schulungsinformationen auf verschiedene Flash-Filme zu verteilen und separat aufzurufen. Hier kann der Anwender individuell entscheiden, welchen Film er ansehen möchte.

Wenn Sie nachträglich einen Flash-Film zu Ihrer Präsentation hinzufügen möchten, können Sie dies ganz einfach tun, indem Sie die HTML-

Abbildung 6 ▸
Set Flash by list: Einstellungsmöglichkeiten

Seite bearbeiten. Es ist nicht notwendig, einen der bereits enthaltenen Flash-Filme aus diesem Grund zu überarbeiten.

Leider hat dieses Verhalten einen entscheidenden Nachteil – ähnlich wie beim Load Movie-Verhalten kann dieses Verhalten nicht vom Netscape Browser dargestellt werden. Nur der Microsoft Internet Explorer ist in der Lage, dieses Verhalten korrekt darzustellen.

Weisen Sie der Grafik »Zoom« ein neues Verhalten zu. Hier wird die Auswahlliste dem Flash-Film zugewiesen. Das Verhalten Set Flash by List wirkt sich direkt auf die Darstellung des Flash-Films aus, ohne dass weitere Verhalten zugewiesen werden müssen. Hinweis: Dieses Verhalten funktioniert nur im Microsoft Internet Explorer.

Erweiterte Formularprüfungen in der Übersicht

Mit den erweiterten Formularprüfungen für Flash 5 haben Sie die Möglichkeit, Eingaben, die in Ihren Flash-Formularen gemacht werden, gleich im Browser des Anwenders auf Richtigkeit hin zu prüfen. Da die gesamte Logik für die gewünschten Prüfungen in Ihre HTML-Seite eingebunden ist, kann eine Prüfung also lokal im Browser des Anwenders stattfinden.

Der Vorteil hierbei liegt darin, dass die Eingaben des Anwenders nicht zuerst zu Ihrem Server übertragen werden müssen, damit hier ein Prüfung stattfinden kann. Sie sparen also die Zeit für die Übertragung zu Ihrem Server und wieder zurück zum Anwender.

Damit die Formularprüfung lokal im Browser des Anwenders stattfinden kann, müssen Sie in Ihrem HTML-Dokument für jedes in Ihrem Flash-Film enthaltene Eingabefeld ein passendes verstecktes Formulareingabefeld einfügen.

◄ Abbildung 7
Die Einstellungsoptionen im Fenster Textoptionen

Das Dreamweaver-Dokument vorbereiten

Erstellen Sie in Dreamweaver ein neues Dokument oder öffnen ein bereits bestehendes Dokument, in das Sie die Formularprüfung einfügen möchten.

Fügen Sie nun ein Formular ein. Geben Sie dem Formular im Fenster EIGENSCHAFTEN einen eindeutigen Namen. Erstellen Sie innerhalb dieses Formulars versteckte Eingabefelder, und geben Sie jedem Eingabefeld einen eindeutigen Namen. Achten Sie darauf, dass jedes Eingabefeld einen anderen Namen trägt, da ansonsten die Prüfung nicht korrekt durchgeführt werden kann.

Fügen Sie nun Ihren Flash-Film in die so vorbereitete HTML-Datei ein. Es ist nicht zwingend notwendig, dass der verwendete Flash-Film innerhalb des HTML-Formulars eingefügt wird. Wichtig ist in diesem Zusammenhang, dass die erweiterte Formularprüfung nur innerhalb einer HTML-Seite durchgeführt und nicht übergreifend auf andere Frames eingesetzt werden kann. Weisen Sie nun Ihrem Flash-Film einen eindeutigen Namen zu.

Nun können Sie im Fenster VERHALTENSWEISEN mittels des PLUS-Buttons ein Ereignis hinzufügen. Wählen Sie aus dem Menüeintrag ADVANCED FORM VALIDATIONS das gewünschte Ereignis aus. Dreamweaver öffnet anschließend ein Dialogfenster, über das die einzelnen Einstellungen für die Ereignisse eingestellt werden, und fügt anschließend das ausgewählte JavaScript in Ihre HTML-Seite ein.

Den Flash-Film vorbereiten

Damit Ihr Flash-Film die JavaScript-Funktionen innerhalb Ihrer HTML-Seite aktivieren kann, müssen Texteingabefelder in Ihrem Flash-Film enthalten sein.

Erstellen Sie ein Textfeld in Flash und weisen ihm im Fenster TEXTOPTIONEN die in der Abbildung 7 gezeigten Einstellungen zu.

Achten Sie darauf, dass Sie hier jedem Textfeld in Ihrem Flash-Film einen eindeutigen und nur einmal verwendeten Namen zuweisen.

Erstellen Sie anschließend in Ihrem Flash-Film einen Button, der als SENDEN-Button fungieren soll. Weisen Sie dem Button das folgende ActionScript als Objektaktion zu:

```
on (press) {
  GetURL ("javascript:FDK_setFormText('form1','John','" add Textfeld
add "'); FDK_setFormText('form1','Gretchen','" add Textfeld add
"');");
}
on (release) {
  GetURL ("javascript:FDK_Validate('form1','false,true',
'Die Anfrage konnte nicht bearbeitet werden\\n\\n');");
}
```

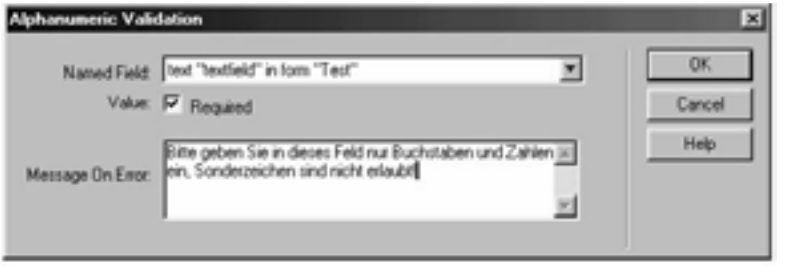
In diesem Beispiel wurde beim Drücken des SENDEN-Buttons die Funktion `FDK_setFormText` verwendet und zusätzlich die Funktion `FDK_Validate`, um die gemachten Eingaben im Browser des Anwenders zu überprüfen.

Die Funktion `FDK_Validate` verfügt über die folgenden vier Parameter:

- `FormName`: Gibt den Formularnamen innerhalb der HTML-Seite an.
- `stopOnFailure`: Wird mit den Werten 0 und 1 ein- bzw. ausgeschaltet.
- `AutoSubmit`: Wird mit den Werten 0 und 1 ein- bzw. ausgeschaltet.
- `ErrorHeader`: Enthält den von Ihnen bestimmten Text für mögliche Fehlermeldungen, die während der Anwendung auftreten können.

Alphanumerische Prüfung

Mit dieser Funktion können Sie prüfen, ob die gemachte Eingabe alphanumerische Zeichen enthält. Gibt der Anwender Sonderzeichen in dieses Feld ein, wird der von Ihnen eingegebene Text in einer Fehlermeldung angezeigt.



◀ Abbildung 8
Die alphanumerische Prüfung

Um diese Funktion in Ihr HTML-Dokument einzufügen, führen Sie die folgenden Schritte aus. Diese Schritte werden exemplarisch erläutert. Alle übrigen Funktionen werden analog ausgeführt.

Wählen Sie den Body-Tag Ihrer HTML-Seite aus, klicken Sie nun im Fenster Verhaltensweisen das Untermenü Advanced Form Validations aus und wählen die Funktion Alphanumeric Validation aus.

In dem sich nun öffnenden Dialogfenster geben Sie den Namen des gewünschten Formularfeldes ein. Stellen Sie mittels des Ankreuzfeldes ein, ob eine Eingabe zwingend notwendig ist oder nicht und geben einen Text für die Fehlermeldung ein, die angezeigt wird, wenn die Prüfung auf einen Fehler stößt.

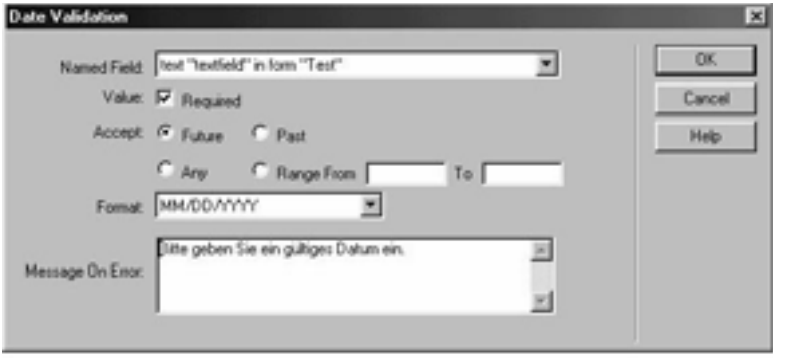
Klicken Sie abschließend auf OK, Dreamweaver fügt nun das ausgewählte JavaScript in Ihre HTML-Seite ein. Als auslösende Handlung (Event) fügt Dreamweaver das `onLoad-Event` in die HTML-Seite ein, ändern Sie diese Einstellung nicht.

Kreditkarten

Mit dieser Funktion können Sie prüfen, ob die gemachten Angaben einer gültigen Kreditkartennummer entsprechen. Die Funktion entfernt automatisch zusätzlich eingegebene Leerzeichen und Querstriche.

Bei der hier durchgeführten Prüfung handelt es sich ausschließlich um eine Plausibilitätsprüfung der Kreditkartennummer. Alle Kreditkartennummern werden auf der Basis einer mathematischen Formel vergeben, die hier überprüft wird. Es wird nicht geprüft, ob die angegebene Kreditkartennummer noch gültig ist oder ob das dazugehörige Konto noch existiert. Hierzu sind zusätzliche Funktionen notwendig, die für die verschiedenen Kreditkarten angeboten werden.

Abbildung 9 ▸
Formularprüfung
Datum



Prüfung Datum

Mit dieser Funktion können Sie prüfen, ob der angegebene Wert einem gültigen Datum entspricht. Hierbei ergeben sich sogar vielseitige Einstellungsmöglichkeiten, die den Einsatz dieser Funktion sehr variabel machen.

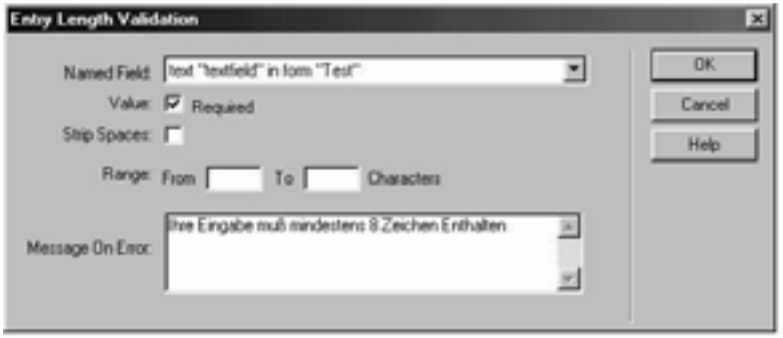
Sie können diese Funktion zum Beispiel dazu verwenden, um zu prüfen, ob das angegebene Datum in der Zukunft oder in der Vergangenheit oder innerhalb eines vorbestimmten Zeitraums liegt.

Auch das Format, in dem das Datum eingegeben werden soll, kann von Ihnen über ein Drop-down-Menü bestimmt werden.

In dem sich öffnenden Dialogfenster geben Sie den Namen des gewünschten Formularfeldes ein. Legen Sie fest, ob das anzugebende Datum in der Zukunft oder in der Vergangenheit liegt, oder bestimmen Sie einen Zeitraum, in dem das Datum liegen soll. Abschließend geben Sie einen Text für die Fehlermeldung ein, die angezeigt wird, wenn die Prüfung auf einen Fehler stößt.

Prüfung E-Mail

Mit dieser Funktion können Sie prüfen, ob die Eingabe einer gültigen E-Mail-Adresse entspricht. Bei der Prüfung wird untersucht, ob die angegebene E-Mail-Adresse ein @-Zeichen und einen Punkt enthält, da dies zwei sehr wichtige Komponenten für eine E-Mail-Adresse sind.



◀ **Abbildung 10**
Formularprüfung Eingabelänge

Prüfung Eingabelänge

Sehr nützlich ist auch diese Prüfung, mit deren Hilfe Sie überprüfen können, ob die gemachte Eingabe eine gewisse Anzahl von Zeichen enthält. Diese Funktion kann zum Beispiel für das Login in geschlossene Benutzergruppen verwendet werden bei dem jeder Anwender ein Login mit einer vorbestimmten Anzahl von Zeichen eingeben muss. Die Eingabe kann auch variabel eingestellt werden, so dass Eingaben mit einer Anzahl von Zeichen »von ... bis ...« möglich sind.

Um diese Funktion in Ihrem HTML-Dokument einzufügen, führen Sie die folgenden Schritte aus (siehe Abbildung 10).

Wählen Sie die Funktion Entry Length Validation aus. Stellen Sie hier das gewünschte Eingabeformat für die Prüfung ein. In dem sich nun öffnenden Dialogfenster geben Sie den Namen des gewünschten Formularfeldes ein. Mit dem Ankreuzfeld Strip Spaces können Sie einstellen, dass automatisch eingegebene Leerzeichen entfernt werden sollen.

Im Bereich Range können Sie einstellen, dass zum Beispiel Eingaben mit einer Länge von zehn bis fünfzehn Zeichen akzeptiert werden. Abschließend geben Sie einen Text für die Fehlermeldung ein, die angezeigt wird, wenn die Prüfung auf einen Fehler stößt. Klicken Sie abschließend auf OK, Dreamweaver fügt nun das ausgewählte JavaScript in Ihre HTML-Seite ein. Als auslösende Handlung (Event) fügt Dreamweaver das `onLoad-Event` in die HTML-Seite ein, ändern Sie diese Einstellung nicht.

Prüfung Fließkomma

Diese Funktion ermöglicht es Ihnen zu prüfen, ob die Eingabe eine Fließkommazahl enthält oder nicht. Diese Funktion kann sehr nützlich bei der Übergabe von Werten sein, die Nachkommazahlen enthalten.

Wählen Sie die Funktion Floating Point Validation aus.

In dem sich nun öffnenden Dialogfenster können Sie im Ankreuzfeld Any einstellen, dass alle eingegebenen Zahlen akzeptiert werden sollen. Im Bereich Range From können Sie einstellen, dass Eingaben mit einer Länge von zum Beispiel acht bis zehn Zeichen akzeptiert werden. Abschließend geben Sie einen Text für die Fehlermeldung ein, die angezeigt wird, wenn die Prüfung auf einen Fehler stößt.

Prüfung Ganzzahl

Analog zur vorhergegangenen Prüfung können Sie mit dieser Funktion prüfen, ob die eingegebene Zahl eine Ganzzahl ist. Auch hier liegt der besondere Nutzen in der Übergabe von Zahlenwerten, die überprüft werden müssen.

Wählen Sie die Funktion Integer Validation aus.

In dem sich nun öffnenden Dialogfenster können Sie im Ankreuzfeld Any einstellen, dass alle eingegebenen Zahlen akzeptiert werden sollen. Im Bereich Range From können Sie einstellen, dass Eingaben mit einer Länge von zum Beispiel acht bis zehn Zeichen akzeptiert werden. Abschließend geben Sie einen Text für die Fehlermeldung ein, die angezeigt wird, wenn die Prüfung auf einen Fehler stößt.

Prüfung internationales Telefon

Mit dieser Funktion können Sie prüfen, ob die Eingabe eine internationale Telefonnummer enthält. Hierbei wird natürlich nicht überprüft, ob die eingegebene Nummer auch vergeben ist. Eine internationale Telefonnummer enthält sechs oder mehr Zahlen, was mit dieser Funktion überprüft werden kann. Bindestriche und Klammern werden automatisch von dieser Funktion entfernt.

Wählen Sie die Funktion International Phone Validation aus.

Vergleichsprüfung

Mit dieser Funktion können Sie zum Beispiel Passworteingaben prüfen. Hierbei kommen zwei Eingabefelder zum Einsatz, die vom prüfenden Skript miteinander verglichen werden. Stimmen die Eingaben nicht überein, wird eine Fehlermeldung eingeblendet. Stimmen beide Eingaben exakt überein, kann zum Beispiel ein Login in einen sicheren Bereich Ihrer Website eingeleitet werden.

Wählen Sie die Funktion Like Entry Validation aus.

In dem sich nun öffnenden Dialogfenster können Sie mit dem Ankreuzfeld Case Sensitive einstellen, ob die Prüfung Groß- und Kleinbuchstaben beachten soll.

Prüfung Formatvorgabe

Mit dieser Funktion können Sie auch komplexe Logins realisieren. Sie können hier individuelle Passwortabfragen einstellen, die nicht das Passwort selbst, aber die Konstellation der eingegebenen Zeichen prüft. Diese Funktion kann auch dazu eingesetzt werden, um zu prüfen, ob die gemachte Eingabe für das Neuanlegen eines Passwortes den vorgegebenen Bedingungen entspricht.

Wählen Sie die Funktion Mask Validation aus. In diesem Beispiel setzt sich die Eingabe aus zwei Buchstaben, zwei Zeichen nach Wahl des Anwenders und zwei Zahlen zusammen. Im Eingabefeld Mask legen Sie fest, wie sich das Passwort zusammensetzen soll.

Prüfung leere Eingabe

Diese Funktion bietet Ihnen die Möglichkeit zu prüfen, ob das Eingabefeld noch leer ist. Häufig lassen Anwender im Internet aus Unachtsamkeit wichtige Eingabefelder aus, was bei der Auswertung der Daten oft zu Verfälschungen führt. Setzen Sie diese Funktion ein, um den Anwender auf die benötigte Eingabe hinzuweisen.

Wählen Sie die Funktion Nonblank Validation aus.

Prüfung Radio-Button

Damit Sie auch prüfen können, ob ein Optionsschalter aus einer Gruppe ausgewählt wurde, steht Ihnen diese Funktion zur Verfügung. Um diese Funktion in Ihr HTML-Dokument einzufügen, wählen Sie die Funktion Radio Button Validation aus. In dem sich nun öffnenden Dialogfenster geben Sie den Namen der Gruppe Ihrer Optionsschalter ein, die überprüft werden sollen.

Prüfung Listenauswahl

Analog zur Prüfung der Optionsschalter können Sie mit dieser Funktion prüfen, ob der Anwender einen Listeneintrag aus einer Auswahlliste ausgewählt hat.

Wählen Sie die Funktion Selection Made in List Validation aus.

Prüfung Sozialversicherungsnummer

Ähnlich wie bei Kreditkartennummern werden Sozialversicherungsnummern nach einem bestimmten Schlüssel vergeben. Mit dieser Funktion können Sie prüfen, ob die Eingabe eine gültige Sozialversicherungsnummer enthält. Um diese Funktion in Ihr HTML-Dokument einzufügen, führen Sie die folgenden Schritte aus. Natürlich ist diese Funktion besonders in den USA interessant, wo sehr viel mit Sozialversicherungsnummern gearbeitet wird.

Wählen Sie die Funktion Selection made in List Validation aus.

Prüfung Uhrzeit

Natürlich darf eine Prüfung für die Eingabe einer Uhrzeit in dieser Kollektion nicht fehlen. Die Funktion Time Validation ermöglicht es, eine Eingabe auf Richtigkeit hin zu überprüfen. Die Funktion geht davon aus, dass Minuten mit angegeben werden.

Damit die Prüfung auch erfolgreich durchgeführt werden kann, ist es notwendig, dass die amerikanischen Kürzel a.m. und p.m. verwendet werden. Die Funktion unterstützt eine Vielzahl von Eingabemöglichkeiten, um die Eingabe nicht zu stark zu reglementieren. Folgende Eingaben sind möglich: am, a.m., AM, A.M., pm, p.m., PM, P.M.

Wählen Sie die Funktion Time Validation aus.

URL-Prüfung

Diese Funktion erlaubt es uns zu prüfen, ob die Eingabe eine gültige URL enthält. Auch diese Funktion wird immer häufiger im Internet abgefragt und muss dementsprechend überprüft werden. Spätestens beim Anmelden Ihrer Domain in einer Suchmaschine werden Sie mit einer solchen Abfrage konfrontiert. Das hier enthaltene JavaScript kennt allerdings weit mehr Formate als das gängige HTTP-Format. Welche das genau sind, sehen wir uns weiter unten genauer an.

Wählen Sie die Funktion URL Validation aus.

In dem sich nun öffnenden Dialogfenster wählen Sie das Textfeld aus, das überprüft werden soll. Mit dem Kontrollkästchen Value können Sie festlegen, ob die Eingabe zwingend notwendig ist. Abschließend geben Sie einen Text für die Fehlermeldung ein, die angezeigt wird, wenn die Prüfung auf einen Fehler stößt.

Die Prüffunktion unterstützt die folgenden Formate:

```
ftp://                  http://
javascript://           file://
gopher://               https://
mailto://               rlogin://
shttp://                snews://
telnet://               tn3270://
swais://
```

Prüfung US-Telefon

Diese Funktion ermöglicht es zu prüfen, ob die Eingabe eine gültige US-amerikanische Telefonnummer enthält. Eine US-amerikanische Telefonnummer enthält zwischen sieben und zehn Zahlen. Die Funktion ist nicht in der Lage zu prüfen, ob die eingegebene Zahl auch tatsächlich vergeben wurde. Bindestriche und Klammern werden von der Funktion automatisch entfernt.

Wählen Sie die Funktion URL Validation aus.

Prüfung Postleitzahlen

Analog zur Prüfung US-amerikanischer Telefonnummern bietet sich durch diese Funktion die Möglichkeit, US-amerikanische Zip-Codes, also Postleitzahlen zu prüfen. US-amerikanische Zip-Codes setzen sich aus fünf- bis neunstelligen Zahlencodes zusammen.

Wählen Sie die Funktion URL VALIDATION aus.

Nehmen Sie hier Ihre Einstellungen für die Postleitzahlenprüfung vor.

Browser-Skripts für Flash in der Übersicht

Die Browser-Skripts für Flash 5 sind eine Sammlung von JavaScripts, welche die Funktionalität Ihrer Flash-Filme zusätzlich erweitern. Sie können diese JavaScripts aus Ihrem Flash-Film heraus aufrufen, um sicher diese Funktionen zu Nutze zu machen. Für die folgenden Übungen sollten Sie in Flash Aktionen zuweisen und ein Grundverständnis im Umgang mit JavaScript haben sowie mit Macromedia Dreamweaver arbeiten können.

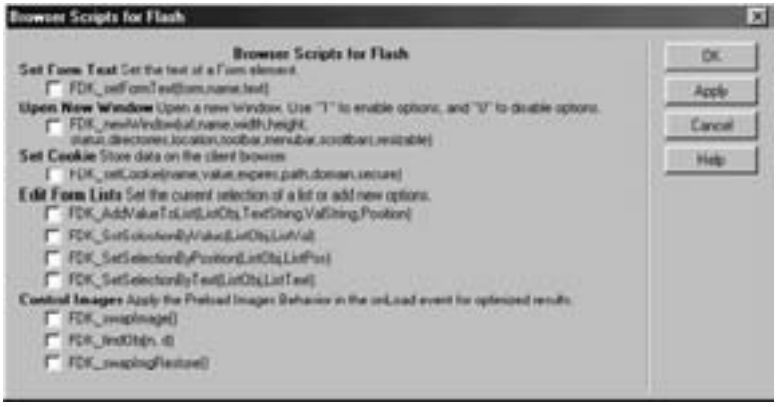
Die Browser-Skripts für Flash ermöglichen es Ihnen, zusätzliche Browserfenster zu öffnen, Cookies zu speichern, Grafik-Rollover-Effekte zu erzeugen sowie Einträge in HTML-Listen zu machen und aus einer Liste oder einem Menü eine Auswahl zu treffen.

Sicher erwarten Sie nun, dass es aufwändig oder kompliziert ist, diese Browser-Skripts in Ihr HTML-Dokument einzufügen, doch weit gefehlt. Starten Sie Macromedia Dreamweaver, und erstellen Sie ein neues Dokument. Speichern Sie das Dokument auf Ihrer Festplatte.

Wählen Sie nun aus dem Menü BEFEHLE den Menüpunkt BROWSER-SKRIPTS FOR FLASH aus. In dem sich nun öffnenden Dialogfenster können Sie auswählen, welche Browser-Skripts Sie in Ihr HTML-Dokument einfügen möchten. Benutzen Sie hierzu einfach die Ankreuzfelder.

Hier erhalten Sie auch einen ersten Überblick über die Parameter, die für die verschiedenen Browser-Skripts benötigt werden. Da jedes der hier gezeigten Browser-Skripts die Dateigröße Ihrer HTML-Seite vergrößert, sollten Sie Skripten, die Sie nicht verwenden möchten, nicht in Ihr HTML-Dokument einfügen.

Wählen Sie in diesem Fenster die gewünschten Browser-Skripts aus. Zurzeit steht leider nur eine englische Version des JavaScript Integration Kit für Flash 5 zur Verfügung. Ob es in absehbarer Zeit eine deutsche Version geben wird, steht leider noch nicht fest!



◀ Abbildung 11
Browser-Scripts für Flash

Set cookie from Flash Movie

Mit dieser Funktion können Sie über die `GetURL`-Aktion den Wert von Cookies einstellen. Dieses eingestellte Cookie kann später dazu genutzt werden, Ihren Flash-Film zu steuern.

Die Funktion `FDK_setCookie` hat insgesamt sechs Parameter, wobei mindestens zwei verwendet werden müssen, damit die Funktion arbeiten kann.

So stellen Sie das Cookie ein:

Weisen Sie in Flash 5 einem Button das folgende ActionScript zu:

```
on (press) {
   GetURL ("javascript:FDK_setCookie('Flashcookie', '1','1/1/
2005','','','');");
}
```

Geben Sie mindestens die ersten zwei der folgenden sechs Parameter an, damit die Funktion ausgeführt wird.

- `Name`: Ist der Name des verwendeten Cookies.
- `Value`: Der Wert, der dem Cookie zugewiesen werden soll.

Die folgenden Parameter können optional verwendet werden:

- `Expires`: Ist das Datum, an dem das Cookie ungültig wird.
- `Path`: Hier können Sie eine URL angeben.
- `Domain`: Hier können Sie eine vollständige Domain angeben.
- `Secure`: Lässt die Werte 0 und 1 zu. Der Wert 1 zeigt an, dass Sie eine SSL-Leitung verwenden, wodurch das Cookie verschlüsselt wird.

Fügen Sie in Dreamweaver nun aus dem Menü Befehle den Eintrag Browser-Skripts for Flash aus und kreuzen das Skript FDK_setCookie an.

Fügen Sie anschließend den fertigen Flash-Film in Ihr HTML-Dokument ein.

Open new Window

Mit dieser Funktion können Sie von Ihrem Flash-Film aus ein neues Browser-fenster öffnen. Auch diese Funktion wird mit der Aktion GetURL verwendet.

Weisen Sie einem Button in Ihrem Flash-Film das folgende Action-Script zu:

```
on (press) {
   GetURL ("javascript:FDK_newWindow('meineSeite.html',
'Player','430','320','','', '', '', '', '', '1',);");
}
```

Die Funktion verfügt über die folgenden elf Parameter:

- `url`: Gibt eine URL der Datei an, die im neuen Fenster angezeigt wird.
- `name`: Der Name für das neue Browserfenster.
- `width`: Die Breite für das neue Browserfenster.
- `height`: Die Höhe für das neue Browserfenster.
- `status`: Mit den Werten 0 und 1 kann der Statusbalken ein- und ausgeblendet werden.
- `directories`: Mit den Werten 0 und 1 kann die Anzeige der persönlichenToolbar-Balken in den Browsern gesteuert werden.
- `location`: Mit den Werten 0 und 1 kann die Anzeige des Adressbalkens inden Browsern gesteuert werden.
- `menubar`: Mit den Werten 0 und 1 kann die Anzeige des Menüs in den Browsern gesteuert werden.
- `scrollbars`: Mit den Werten 0 und 1 kann eingestellt werden, ob das neue Fenster scrollen kann oder nicht.
- `resizable`: Mit den Werten 0 und 1 kann eingestellt werden, ob das neue Fenster in seiner Größe verändert werden kann oder nicht.

Fügen Sie in Dreamweaver nun aus dem Menü Befehle den Eintrag Browser-Skripts for Flash aus und kreuzen das Script FDK_newWindow an.

Fügen Sie anschließend den fertigen Flash-Film in Ihr HTML-Dokument ein.

ImageRollover from Flash Movie

Mit dieser Funktion können Sie Grafiken innerhalb Ihrer HTML-Seiten austauschen, sobald der Mauszeiger über vordefinierte Regionen innerhalb Ihres Flash-Films positioniert wird. Hierdurch lassen sich sehr nützliche Rollover-Effekte erzeugen.

Damit die Funktion ausgehend von Ihrem Flash-Film auf die Grafik zugreifen kann, muss der Grafik in Dreamweaver ein Name im Fenster Verhaltensweisen zugewiesen werden.

Weisen Sie einem Button in Ihrem Flash-Film das folgende ActionScript zu:

```
on (press) {
   GetURL ("javascript:FDK_swapImage('OrgLogo','','logo2.gif',1);");
}
on (rollOut) {
   getURL ("javascript:FDK_swapImgRestore();");
}
```

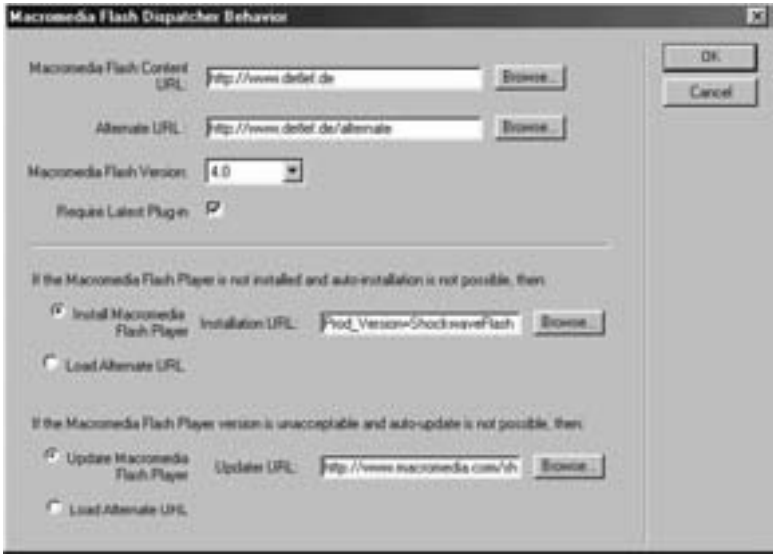
Wie Sie sehen, werden hier gleich zwei Funktionen verwendet. Die erste Funktion tauscht die ursprünglich angezeigte Grafik gegen eine zweite Grafik aus. Die zweite Funktion stellt nach dem Rollover den Ursprungszustand wieder her und zeigt die erste Grafik an.

Die Funktion verfügt über die folgenden vier Parameter:

- `name`: Der Name des Bildes, das ausgetauscht werden soll.
- `empty`: Dieser Parameter wird nicht benutzt und bleibt leer.
- `path`: Der vollständige Pfad zum Austauschbild.
- `restore`: Geben Sie hier eine 1 ein, damit das ursprüngliche Bild nach dem Rollover wieder angezeigt wird.

Fügen Sie in Dreamweaver nun aus dem Menü Befehle den Eintrag Browser-Skripts for Flash aus und kreuzen die Skripten FDK_swapImage, FDK_swapImgRestore und FDK_findObj an. Fügen Sie anschließend den fertigen Flash-Film in Ihr HTML-Dokument ein.

Abbildung 12 ▶
Macromedia Flash Dispatcher Behavior



Macromedia Flash Dispatcher Behavior

Lassen Sie uns nun noch im Folgenden einen Blick auf das Macromedia Flash Dispatcher Behavior werfen, mit dessen Hilfe wir unsere HTML-Seiten optimal vorbereiten können.

Jeder Flash-Anwender fragt sich spätestens nach der Erstellung seines ersten Flash-Films, wie er sicherstellen kann, dass der Anwender im Internet seinen Flash-Film richtig angezeigt bekommt. Genau hier setzt der Dispatcher an. Mit diesem JavaScript-Verhalten prüfen wir, ob der verwendete Browser in der Lage ist, den enthaltenen Flash-Film richtig darzustellen. Auch diese Arbeiten nehmen wir in Macromedia Dreamweaver vor.

Das Macromedia Flash Dispatcher Behavior ist fester Bestandteil des JavaScript Integration Kit für Flash 5. Um das Dispatcher-Verhalten in eine HTML-Seite einzubinden, gehen Sie wie folgt vor:

Legen Sie ein neues HTML-Dokument an und speichern es unter einem neuen Namen ab. Wählen Sie den Body-Tag aus und im Fenster Verhaltensweisen den Menüeintrag Macromedia Flash Dispatcher Behavior.

In dem sich nun öffnenden Dialogfenster können Sie einstellen, unter welcher URL Ihre Flash-Dateien erreichbar sind und unter welcher URL alternative, sprich Flash-freie Dateien abgerufen werden können.

Die verwendete Flash-Version kann angegeben werden und welches Plug-in für Ihre Seiten verwendet werden soll. Ist noch kein Flash-Plug-in

im Browser des Anwenders enthalten, wird der Anwender automatisch zur Website von Macromedia geführt, wo er das Plug-in herunterladen oder, für den Fall, dass ein veraltetes Plug-in installiert ist, ein Update erhalten kann.

Der Flash Dispatcher bietet eine gute Möglichkeit, zu prüfen, ob der Browser des Anwenders bereits mit einem passenden Plug-in ausgestattet ist. Wer sichergehen will, dass der Anwender passend ausgestattet ist, dem empfehle ich unbedingt die Flash Detection der Flashworker, zu finden unter www.flashworker.de (siehe auch Seite 604).

Später im Buch wird auch beschrieben, wie man sich eine eigene baut.

Erweiterung eines Flash-Movies mit dem Kommandozeileninterpreter

Es gibt einfache Möglichkeiten, die Funktionsvielfalt des Players zu erweitern.

▸ Alexander Kutz
www.medienmeile.de

Auslöser, dieses Kapitel zu schreiben, waren die begrenzten Möglichkeiten, mit dem Flash-Player in Windows zu arbeiten. Ich habe versucht, ein PDF-Dokument aus dem Flash-Player zu öffnen, und es klappte erstmals nur mit `GetURL`. Dann wurde das Dokument im Browser geöffnet – zudem im Internet Explorer und nicht in meinem Standardbrowser von Netscape. Von Nachteil ist außerdem, dass sich das Dokument nicht im Vollbildmodus (da nicht direkt im Acrobat Reader) darstellen lässt.

Das Tutorial ist jetzt mit Flash 4-Screenshoots angefertigt, es lässt sich aber problemlos in Flash 5 anwenden.

Schnittstellen zur Erweiterung

Es gibt zwei Befehle, um aus Flash auf eine andere Datei zuzugreifen:

1. `getURL`
   Kann nicht verwendet werden, da der Browser nur extrem begrenzte Zugriffsrechte auf das Dateisystem hat. Programme können hier nicht ausgeführt werden.
2. `FS Command ("exec", "")`
   Wird von uns verwendet. Mit diesem Befehl können Programme gestartet werden. (Dokumente können nicht direkt geöffnet werden.)

◄ Abbildung 1
Aufruf von FS Command

Der Kommandozeileninterpreter

Was ist ein Kommandozeileninterpreter?

Um mit dem Betriebssystem richtig arbeiten zu können, wird eine Benutzeroberfläche gebraucht. Diese kann grafisch sein (**G**raphical **U**ser **I**nterface) – zum Beispiel Windows – oder auch textbasiert. In diesem Fall ist sie kommandozeilenorientiert mit Eingabeaufforderung, z. B. MS DOS oder eine Linux-Shell.

Ein Kommandozeileninterpreter ist ein Programm, das diese Befehle aufnimmt und zum System weiterleitet.

Woher bekomme ich ein solches Programm?

Entweder schreiben Sie ein solches Programm selbst, oder Sie verwenden das Programm von Microsoft. Microsoft hat ein solches Programm für NT veröffentlicht. Es funktioniert auch mit anderen Windows-Versionen und nennt sich ShelExec.exe.

Die Datei gibt es auf diversen FTP-Servern.

Effektive Möglichkeiten

Dieser Kommandozeileninterpreter bietet die Möglichkeit, alle Dokumente mit dem jeweils dafür registrierten Programm zu öffnen, z. B. *.doc-Dokument mit MS Word oder ein *.pdf-Dokument mit dem Adobe Acrobat Reader und nicht über ein Plug-in.

Im Standardbrowser werden Internetseiten geöffnet, eine FTP-Verbindung wird mit dem FTP-Programm geöffnet, und man kann eine E-Mail im Standard-Mail-Client verfassen.

Syntax

Die ShelExec.exe wird in Flash wie ein normales Programm geöffnet:

```
FS Command ("exec", "Shelexec.exe")
```

Damit die ShelExec.exe erkennt, was sie zu erledigen hat, muss es ihr beim Aufruf mitgeteilt werden. Um z. B. www.flashworker.de zu öffnen, geben Sie Folgendes ein:

```
Shelexec.exe http://www.flashworker.de
```

Damit die Parameter direkt mit ausgegeben werden können, sind die Leerstellen in Flash durch die Zeichennummer auszudrücken. Fertig sieht das Ganze dann so aus:

```
FS Command ("exec", "Shelexec.exe"&Chr(9)
&"http://www.flashworker.de")
```

Folgende Parameter werden erkannt:

- ftp://
- http://
- mailto:
- Dateien (*.*)

Pfade

Die Pfade sind immer relativ zur Shelexec.exe. Liegt die zu öffnende Datei im selben Verzeichnis, muss ein .\ vorangestellt werden.

Beispiel:

```
FS Command ("exec", "Shelexec.exe"&Chr(9)& ".\dokumentation.PDF")
```

Die Shelexec.exe öffnet nun die Dateidokumentation.PDF aus demselben Verzeichnis. Mit

```
FS Command ("exec", "Shelexec.exe"&Chr(9)&"test\info.doc")
```

öffnet die Datei info.doc aus dem Verzeichnis test, mit

```
FS Command ("exec", "Shelexec.exe"&Chr(9)&
"mailto:info@medienmeile.de?Subject=Flash-Buch")
```

den Standard-Mail-Client mit der Adresse info@medienmeile.de und dem Betreff »shelexec«.

Links

Detlef Biermann: Autoplay-Funktion für HTML-Dateien von CD-ROM unter Windows. Hier wird die Verwendung der Shelexec beschrieben.
http://www.teamone.de/selfaktuell/artikel/autorun.htm

Ein Programm selbst schreiben

Es ist recht einfach, einen Kommandozeileninterpreter in Visual Basic oder C++ zu schreiben.

Die Vorteile:

- Das eigene Icon kann verwendet werden.
- Die Funktionen können beliebig erweitert werden.
- Ebenso ist es möglich, ein Programm zu schreiben, das den Rechner herunterfahren oder einen Neustart auslösen kann.

Mehr Informationen finden sich hier:

Allgemeine Anleitungen zu einem Kommandozeileninterpreter:
http://www.codeproject.com/system/shelexec.asp

Shell-Programme in C selbst schreiben:
http://www.codeproject.com/shell/

Shellexec in VB schreiben:
http://www.schwider.de/shellexe.htm

Weitere ausführliche Informationen zum CD-Recording:
http://www.brennmeister.com/cdrfaq/

Vor dem Brennen ...

... auf CD sollten Sie die Shelexec.exe per Attribut verstecken.

Weitere nützliche Tools für die Arbeit mit Flash

Im Folgenden bieten wir Ihnen eine Sammlung von Tools, mit denen Flash sehr gut zusammenarbeitet.

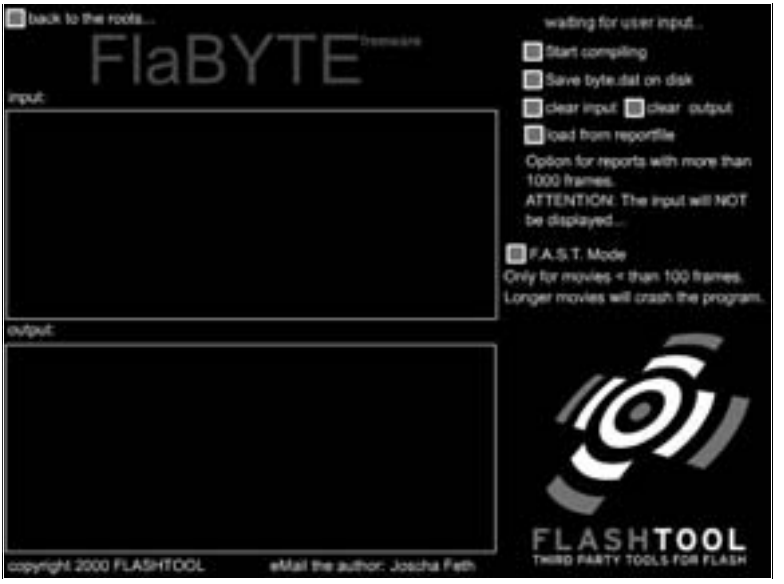
ActionScript und Animationstools

Die Flash-Tools

Gerald Marischka

Auf der Webseite www.flashtool.de stehen Ihnen verschiedene sehr interessante Tools zur Verfügung.

- Das Tool FlaPRINT druckt den Inhalt eines Projektorfensters oder Textfeldes aus.
- Flash Unprotect hebt den Importschutz von SWF-Dateien auf.
- FlaWRITE ist ein Tool, mit dem sich offline (temporäre) Dateien auf die Festplatte schreiben lassen.
- Der FlaBYTE Preloader ermöglicht, dass beim Ladevorgang der Fortschritt nicht nur in % angezeigt wird, sondern auch die Anzahl der bereits geladenen Bytes aufgeführt ist.
- Mit WildSWfX lassen sich in wenigen Minuten komplexe Textanimationen erstellen, für die man in Flash normalerweise Stunden brauchen würde, http://www.wildswfx.com.
- Mit FlaSE kopieren Sie ActionScript problemlos in das Aktionsfenster von Macromedia Flash.



◀ Abbildung 1
Der FlaBYTE Preloader

◀ Abbildung 2
WildSWfX

Abbildung 3 ▶
FlaSE

Timo Meteling ▶
www.intabo.de,
Björn Gromoll,
www.powerflasher.de,
Ronny Hendrichs,
www.hendrichs.de

Liste aller Tools

Hersteller	Software name	Kategorie	Beschreibung	D/E	Preis	Kaufmöglichkeit	System
Ideaworks 3d	Vecta3d Standalone	3D-Vektorgrafik	Erstellt aus verschiedenen 3D-Formaten 3D-Vektorgrafiken im SWF- und AI-Format.	E	$ 74,95	www.buyonet.com/s/b	Windows 95, 98 und NT
Ideaworks 3d	Vecta3d-Max	3D-Vektorgrafik	Erstellt aus 3D Studio Max-Dateien Vektorgrafiken im SWF- und AI-Format.	E	$ 194.95	www.beyond.com	Windows 95, 98 und NT
Electric Rain	Swift 3d 1.0	3D-Vektorgrafik	Erstellt aus EPS-, AI- und 3ds-Dateien 3D-Vektorgrafiken im SWF-Format.	E	$ 139.00	www.swift3D.com	Windows 98, NT und 2000
Martin Fleck	Action Script Editor	Action-Script-Editor	Erleichtert die ActionScript-Programmierung; kann AS-Befehle in Flash importieren.	D	DEM 29	www.flashworker.de	Windows 95, 98 und NT
Grooveware Multimedia	FAST! Flash Action-Script-Tool	Action-Script-Editor	Erleichtert die Action-Script-Programmierung; kann AS-Befehle in Flash importieren.	E	DEM 83	www.swifftools.com/stools/	Windows 95, 98 und NT
St. Clair Software	Autolaunch	Bildschirmschoner	Wandelt Stand-alone-Projektoren von Flash und Director in Bildschirmschoner um.	E	$ 15 Shareware	https://order.kagi.com/cgi-bin	Mac OS

Hersteller	Softwarename	Kategorie	Beschreibung	D/E	Preis	Kaufmöglichkeit	System
St. Clair Software	Autolaunch Pro1	Bildschirmschoner	Wandelt Stand-alone-Projektoren von Flash und Director in Bildschirmschoner um.	E	ab $ 250	https://order.kagi.com	Mac OS
Magic Modules	Buddy Saver	Bildschirmschoner	Erstellt aus Director, Flash oder Authorware-Projektoren Bildschirmschoner.	E	$ 75	www.mods.com.au	Windows 95, 98 und NT
CINOAN Solutions	CINOAN Screen-Saver for Flash Beta	Bildschirmschoner	Erstellt aus Flash-Projektoren Bildschirmschoner.	E	unklar	www.first-base.demon.co.uk	Windows 95, 98 und NT
Flashjester	Creator	Bildschirmschoner	Erstellt aus Flash-Projektoren Bildschirmschoner; zahlreiche Einstellungen und Optionen; gibt es in zwei Versionen: Standard und Pro.	E	ab $ 395	www.flashjester.com/software/creator/	Windows 95, 98 und NT
Flashjester	Entertainer	Bildschirmschoner	Spielt SWF-Dateien als Bildschirmschoner ab.	E	$ 29	http://shareit1.element5.com	Windows 95, 98, NT
Goldshell	Flashforge 5	Bildschirmschoner	Erstellt aus Flash-Stand-alone-Projektoren Bildschirmschoner.	E	$ 60	https://www.regsoft.net	Windows 95, 98, NT
I-D Media AG	Living Screen	Bildschirmschoner	Erstellt aus Flash-Stand-alone-Projektoren Bildschirmschoner.	E	ab $ 149	www.livingscreen.com	Windows 95, 98 und NT
SCORE Multimedia	Screensaver Wizard	Bildschirmschoner	Erstellt aus Director- und Flash-Stand-alone-Projektoren einen Bildschirmschoner.	D	$ 80	http://shareit1.element5.com	Windows 95, 98 und NT
MacSourcery	ScreenTime for Flash	Bildschirmschoner	Erstellt aus Flash-Stand-alone-Projektoren Bildschirmschoner.	E	ab $ 299	www.macsourcery.com/	Windows 95, 98 und NT
Grooveware Multimedia	Screenweaver 1	Bildschirmschoner	Erstellt aus Flash-Dateien Bildschirmschoner.	E	gratis	www.screenweaver.com	Windows 95, 98 und NT
Grooveware Multimedia	Screenweaver 2	Bildschirmschoner	Erstellt aus Flash-Dateien Bildschirmschoner.	E	$ 150	www.swifftools.com/Stools	Windows 95, 98 und NT
David Goldsmith	AnySaver	Bildschirmschoner	Wandelt EXE-Dateien in Bildschirmschoner um, der nicht kommerzielle Einsatz ist kostenlos.	E	gratis	www.dgolds.com	Windows 95, 98 und NT
Jeroen Kessels	Form2Flash	Dynamische Flash-Inhalte	Erstellt aus HTML-Formularen dynamische Flash-Inhalte.	E	gratis	www.kessels.com	Windows NT, Unix

Hersteller	Softwarename	Kategorie	Beschreibung	D/E	Preis	Kaufmöglichkeit	System
Blue* Pacific	Turbine 4	Dynamische Flash-Inhalte	Generiert aus ASP-Scripts dynamische Flash-Inhalte (gibt es in vier Versionen).	E	ab DEM 750	www.blue-pac.com/products/	viele ...
Jeroen Kessels	FlashDB	Dynamische Flash-Inhalte	Eine Datenbank für Flash4 geschrieben in Perl	E	gratis	www.kessels.com	Windows NT, Unix
Macromedia	Generator 2	Dynamische Flash-Inhalte	Erstellt aus diversen Quellen (z. B. Datenbanken) dynamische Flashinhalte	E/D	DEM 9.999	www.softline.de	Windows NT
Oliver Debon	Swift-Generator	Dynamische Flash-Inhalte	Produziert aus eigenen Vorlagedateien dynamische Flash-Inhalte.	E	gratis/ 204 pro Webseite	http://shareit1.element5.com	viele ...
Goldshell	Versiown	Projektortool	Tauscht das Standard-Icon sowie die Dateieigenschaften der Stand-alone-Projektoren aus.	E	DEM 83	www.goldshell.com	Windows 95, 98 und NT
Flashjester	Jugglor	Projektortool	Verschiedene Optionen zum Anpassen der Stand-alone-Projektoren (z. B. Abschalten der rechten Maustaste.).	E	DEM 206	http://navworks.i-us.com	Windows 95, 98 und NT
Flashjester	JTools	Projektortool	Flash-Stand-alone-Projektoren können hiermit Programme starten oder Dateien mit diversen Inhalten aufrufen (z. B. PDF-Dateien).	E	DEM 102	http://shareit1.element5.com	Windows 95, 98 und NT
Flashjester	JAvi	Projektortool	Erweitert die Flash-Stand-alone-Projektoren um die Möglichkeit, AVI-Videos einzubinden.	E	DEM 206	http://shareit1.element5.com	Windows 95, 98 und NT
Flashjester	JPrintor	Projektortool	Erweitert die Flash-Stand-alone-Projektoren um eine frei anpassbare Druckfunktion.	E	DEM 206	http://shareit1.element5.com	Windows 95, 98 und NT
Mories.com	Projector Launcher	Projektortool	Verschiedene Optionen zum Anpassen der Stand-alone-Projektoren (z. B. Abschalten der rechten Maustaste).	E	DEM 144	http://shareit1.element5.com	Windows 95, 98 und NT
Goldshell	Hoolicon	Projektortool	Tauscht das Standard-Icon der Stand-alone-Projektoren aus.	E	DEM 62	www.goldshell.com	Windows 95, 98 und NT
Groove-ware Multi-media	SwiffPEG	Sound-Tool	Konvertiert MP3-Files in SWF-Dateien (gibt es auch als Server, zurzeit in der Betaphase).	E	DEM 52	www.swifftools.com	Windows 95, 98 und NT

Hersteller	Software name	Kategorie	Beschreibung	D/E	Preis	Kaufmöglichkeit	System
Burak KALAYCI	Swifty Utilities	SWF-Tool	Diverse Tools für Flash 4 (z. B. MovieClip Extractor, Unprotector, Layer Extractor usw.).	E	gratis	http://buraks.com/swifty	Windows 95, 98 und NT
Flashjester	Woof	SWF-Tool	Tool zum Anschauen und Kopieren der Flash-Dateien aus dem Browsercache.	E	DEM 60	http://shareit1.element5.com	Windows 95, 98 und NT
Grooveware Multimedia	SWF-Browser 2.9	SWF-Tool	Kann z. B. Bitmaps oder Sounds/Loops aus SWF-Dateien extrahieren.	E	DEM 52	www.swifftools.com	Windows 95, 98 und NT
Oliver Debon	Swift-Inspector	SWF-Tool	Kann z. B. Bitmaps oder Sounds/Loops aus SWF-Dateien extrahieren.	E	gratis	www.swifftools.com	Linux
DJJ Holdings Pty Ltd.	SwiSH	Texteffekte	Erstellt in Sekunden Texteffekte im SWF-Format.	E	DEM 62	https://secure.bmtmicro.com/ECommerce-custom/59200.html	Windows 95, 98 und NT
Manitu Group	URL Action Editor	Action-Script-Editor		E	$ 39.95	http://buraks.com/uae/	Windows 95, 98 und NT
goldshell	Flax	Typotool	Zum Erstellen von Texteffekten.	D, E	$ 40	http://www.flaxfx.com/	Windows 95, 98
Wanpatan Software Lab	Bitbull	screeen saver	Erstellt aus Director- und Flashfiles Screensaver.	E	unklar	http://www.wanpatan.com/bitbull/	Windows 95, 98 und NT
Manitu Group	Action Script Viewer 1.5	Action-Script Viewer	Dieses Tool gibt Einsicht über Anzahl von Buttons Szenen etc. in einer SWF	E	unklar	http://buraks.com/asv/	Windows 95, 98 und NT

Server, Dynamik und Flash

438 Macromedia Generator

438 Wie funktioniert der Generator?
438 Einsatzgebiete
440 Generator-Objektpalette und Generator-Inspektor
441 Generator-Variablen definieren
443 Umgebungsvariablen und Datenformate?
449 Tutorials zur Einstellung der Umgebung des Generators
449 Tutorials zum Generatorobjekt Table und Insert JPEG File
451 Tutorial zum Generatorobjekt Plot
455 Tutorial zum Generatorobjekt Scrolling List
459 Weitere Fähigkeiten von Generator

462 Swift-Generator

462 Was ist der Swift-Generator?
465 Erstellen der Vorlage – Templates oder Flash-Datei?
466 Swift-Variablen in die Vorlage einfügen
468 Das Swift-Skript
480 Kommandozeile oder CGI – online oder offline
481 Swift-Generator auf dem Server installieren
482 Swift-Generator im eigenen Verzeichnis
483 Bekannte Probleme

486 Anwendungsbeispiele für serverseitige Skripts
486 Flash, PHP & MySQL
493 Flash und ASP
496 E-Mail-Versand mit ASP
503 CGI und Flash
508 Gästebuch mit Flash und Perl-CGI
510 Textdateien editieren

516 Flash und XML
516 XML-Einführung
517 Warum XML in Flash?
519 Document Object Model (DOM)
522 Parsen des XML-Dokuments

524 Flash-Newsletter
524 Was ist vor dem Upload zu tun?
526 Datei-Upload

528 Ming – SWFs ohne Flash

530 Einlesen der Uhrzeit und des Datums
532 Die JavaScript-Variante
558 Die Perl-Variante
565 Die PHP-Variante
567 Die Flash 5-Variante

Macromedia Generator

Wir werden uns in diesem Kapitel einen Überblick über die Fähigkeiten und Einsatzgebiete von Generator 2.0 verschaffen.

▸ Hassan Beigi, www.powerflasher.de

Generator ist die hauseigene Lösung für dynamische Flash-Anwendungen von Macromedia. Mit Generator ist der Entwickler in der Lage, Templates mit variablem Inhalt zu erzeugen. Die Templates/Platzhalter werden nach der Installation von Generator nahtlos in die Flash-Entwicklungsumgebung eingefügt.

Mit diesen Platzhaltern kann der Entwickler Grafiken, Text, Variablen, Charts, Sounds und vieles mehr dem Projekt hinzufügen. Die Platzhalter werden bei der Generierung des Flash-Films durch ihren Inhalt ersetzt. Der Inhalt für die Platzhalter kann sowohl aus Textdateien und Datenbanken als auch von Java-Anwendungen, http- und FTP-Adressen stammen.

Generator bietet die ideale Möglichkeit, ohne viel Aufwand dynamische E-Commerce-Anwendungen oder Webseiten mit benutzerspezifischem Inhalt zu erzeugen.

Wie funktioniert der Generator?

Mit Generator wird statt einer SWF-Datei eine SWT-Datei erzeugt. Die SWT-Datei beinhaltet die Platzhalter für den dynamisch zu erzeugenden Inhalt.

Einsatzgebiete

Wir unterscheiden zwei Einsatzmöglichkeiten von Generator: die Onlinegenerierung und die Offline-Generierung.

Online-Generierung

Bei dieser Methode wird die Generator-Template-SWT-Datei statt einer SWF-Datei in die HTML-Datei eingefügt. Dabei ruft der Client-Webbrowser statt einer SWF- eine SWT-Datei auf. Die Webserver-Erweiterung erkennt diese Anfrage, und Generator erstellt aus der SWT-Datei und den Quellen für die Inhalte den Platzhalter – eine SWF-Datei. Schließlich wird die neu erzeugte SWF-Datei wie gewohnt im Client-Browser dargestellt.

Generator ersetzt bei der Bearbeitung des Templates die Generator-Variablen und die Platzhalter durch die ihnen zugewiesenen Inhalte. Der Online-Generator ist eine Erweiterung des Webservers und muss bei der Installation des Webservers mit installiert werden. Die Erläuterungen zur Einrichtung der Servererweiterung würde den Rahmen dieses Kapitels sprengen.

Der Online-Generator eignet sich für Flash-Filme, deren Inhalt bei jedem Aufruf der Seite neu generiert werden muss. Solche Filme könnten unter anderem E-Commerce-Anwendungen, Darstellung von Aktienkursen oder Wetterkarten sein. Dabei wird mit den aktuellsten Daten ein Flash-Film erzeugt.

Für die Generierung müssen HTML-Dokumente und die eingebundenen SWT-Dateien auf dem Server kopiert werden.

Offline-Generierung

Die Offline-Generierung eignet sich für Flash-Filme, deren Inhalt nicht ständig neu generiert werden muss und für jeden Benutzer derselbe ist.

Dabei läuft Generator nicht als eine Servererweiterung und spart so die Ressourcen des Webservers. Man kann den Offline-Generator dafür einsetzen, um die Aktualisierung der Inhalte unkomplizierter zu gestalten. Dabei kommt der Inhalt des Flash-Films nicht direkt aus Flash, sondern aus externen Datenquellen. Somit ist für die Veränderung der Inhalte in Flash keine Flash-Entwicklungsumgebung mehr nötig. Der Kunde kann in so einem Fall den Inhalt der Webseiten selbst verändern und aktualisieren.

Vor jedem Projekt ist zu bedenken, ob man online oder offline generieren möchte und welche Inhalte dynamisch sein müssen. Je mehr dynamische Elemente eingesetzt werden, desto einfacher wird die Wartung der Seiten.

Die Veröffentlichung mit der Offline-Version ist die gleiche wie bei normalen Flash-Projekten. Die offline generierte SWF-Datei wird in das HTML-Dokument eingebunden und auf dem Server kopiert. Die SWT-Dateien müssen bei der Offline-Generierung nicht mit auf dem Server kopiert werden.

Generator 2.0 gibt es als Developer- und Enterprise-Version. Für die Onlinegenerierung empfiehlt sich die Enterprise-Version. Die Developer-Version eignet sich weniger für die Online-Generierung, da die Developer-Version immer nur eine SWT-Datei gleichzeitig erstellen kann. Generator gibt es für verschiedene Systeme und Plattformen.

Generator-Objektpalette und Generator-Inspektor

Nach der Installation der Generator-Entwicklungswerkzeuge (Authoring Extensions) stehen uns in der Flash-Entwicklungsumgebung die Generator-Objektpalette und der Generator-Inspektor zur Verfügung. In der Generator-Objektpalette befinden sich die verschiedenen Platzhalter. Diese Platzhalter werden auf der Bühne positioniert. Jeder Platzhalter und jede Instanz aus der Flash-Bibliothek hat nun Generator-Eigenschaften, die mit dem Generator-Inspektor bearbeitet werden können. Jeder Platzhalter hat seine eigenen spezifischen Eigenschaften, die wir später in diesem Kapitel näher erläutern werden.

Es stehen uns die in Abbildung 1 gezeigten, vordefinierten Platzhalter zur Verfügung.

Pfad der Symbole in der Flash-Bibliothek angeben

Um mit den Generator-Objekten arbeiten zu können, muss man in der Lage sein, Symbole aus der Flash-Bibliothek zu referenzieren (die Position angeben). Wir haben die in Abbildung 2 gezeigte Flash-Bibliothek in einem Projekt:

- Das Symbol `menue` wird einfach mit `menue` referenziert.
- Die Symbole `graphic1`, `bild` und `text` werden wie folgt referenziert: `graphics/graphic1` , `style/bild` und `style/text`.

◂ **Abbildung 1**
Die Platzhalter

◂ **Abbildung 2**
Die Bibliothek

Generator-Variablen definieren

Generator-Variablen sind Platzhalter für Werte, die aus externen Datenquellen bezogen werden. Generator-Variablen werden in {} eingeschlossen. Jeder in {} eingeschlossene Text definiert eine Generatorvariable und wird bei der Generierung ersetzt.

Generator-Variablen können an verschiedenen Stellen auftreten.

- **Generator-Variablen auf der Bühne**
  Wenn eine Generatorvariable auf der Bühne positioniert werden soll, erstellen wir ein statisches Textfeld auf der Bühne. In dem Textfeld tragen wir den Namen der Generatorvariablen ein. Wenn unsere Variable `genvar_1` heißen soll, dann tragen wir in dem statischen Textfeld `{genvar_1}` ein. Wir weisen diesem statischen Textfeld die uns bekannten Eigenschaften für Textgröße, Texttyp, Farbe etc. zu. Der Wert von `genvar_1` wird dann mit diesen Eigenschaften dargestellt.

- **Generator-Variablen in ActionScripts**
  Wir können in ActionScripts mit Generator-Variablen arbeiten. So können wir dynamische ActionScripts schreiben. Dazu ein kleines Beispiel. Wir gehen davon aus, dass wir drei Generator-Variablen {var1}, {var2} und {target} definiert haben. Mit der folgenden Anweisung können wir feststellen, ob der Wert von `var1` gleich dem Wert von `var2` ist, und eine Aktion durchführen:

```
if ("{var1}" == "{var2}") {
    gotoAndStop ("{target}");
}
```

  Beachte, dass die Generator-Variablen in Anführungszeichen eingeschlossen sein müssen.
  Es ist nicht möglich, als Ziel eine Bildnummer als Generator-Variable anzugeben.
  Die Variable `{target}` muss einen Bildnamen als Wert besitzen.
- **Generator-Variablen als Wert für die Eigenschaft eines Platzhalters**
  Es ist möglich, im Generator-Inspektor eine Generator-Variable statt einen fixen Wert für eine Eigenschaft eines Platzhalters anzugeben.

Dazu ein kleines Beispiel. Wir positionieren einen »Insert JPEG File«-Platzhalter auf die Bühne und importieren die Variablen `{bildquelle}` und `{qualität}` als Umgebungsvariablen des Generators aus der Textdatei params.txt (siehe »Was ist die Funktion der externen Datenquellen?«). params.txt ist wie folgt aufgebaut:

```
Name, Value
bildquelle, "001.jpg"
qualität, "50"
```

Für die Eigenschaft »File Name« des JPEG-Platzhalters geben wir die variable `{bildquelle}` an. Und für die Eigenschaft »JPEG Quality« geben wir die variable `{quality}` an. Wenn wir jetzt generieren, wird der JPEG-Platzhalter durch das Bild 001.jpeg mit der Qualität 50 % ersetzt.

Mit dieser Methode ist es uns möglich, für die nächste Generierung die Textdatei zu bearbeiten, um das Bild auszutauschen und dessen Anzeigequalität neu zu definieren. Die Anwendungsmöglichkeiten werden in praktischen Beispielen näher erläutert.

◄ Abbildung 3
Umgebungsvariablen des Generators

Umgebungsvariablen und Datenformate?

Nach der Installation von Generator erscheint in der Flash-Entwicklungsumgebung, oben in der Leiste neben dem Symbol SZENE BEARBEITEN, das blaue Symbol UMGEBUNGSVARIABLEN DES GENERATORS.

Mit einem Klick auf dieses Symbol öffnet sich das Fenster zur Einstellung der Umgebung. Hier können wir der Hauptzeitleiste Datenquellen zuweisen oder direkt Variablen definieren.

Ein Beispiel: Wenn wir dort params.txt angeben, werden die definierten Variablen in params.txt zu Generator-Variablen des Projekts. Mehrere Datenquellen werden durch Semikolon voneinander getrennt. Man kann nur der Hauptzeitleiste des Projekts oder der Hauptzeitleiste eines MovieClips Umgebungsvariablen zuweisen. Grafik- und Tastensymbole sind davon ausgeschlossen.

Generator-Variablen und Generator-Objekte/-Platzhalter beziehen ihren Inhalt aus den verschiedensten externen Datenquellen.

Wir erläutern lediglich die zwei wichtigsten Datenquellen:

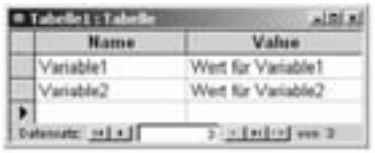
- Textdateien
- Datenbanken

Dabei muss die Datenquelle so formatiert sein, dass Generator die Daten auswerten kann.

Variablen können in zwei verschiedenen Formaten zur Verfügung gestellt werden:

1. Datenformat für Name/Wert
2. Datenformat für Spaltenname/Spaltenwert

Namen-/Wertvariablen können nur einen Wert pro Variable besitzen, wogegen Spaltennamen-/Spaltenwertvariablen mehrere Werte besitzen können.



▲ Abbildung 4
Datenbanktabelle

Abbildung 5 ▶
Datenquellen (ODBC)

Das Format Spaltenname/Spaltenwert wird von den meisten Platzhaltern (Plot, Table etc.) benötigt. Mit diesem Format können wir den Platzhaltern Datenverbunde (Records, mehrere Datensätze) zuweisen.

Datenformat für Name/Wert

Wenn es sich um eine Textdatei als externe Datenquelle handelt, erstellen wir eine Textdatei params.txt (Name ist beliebig). Die erste Zeile wird wie folgt erstellt:

```
Name, Value
```

Auf diese Zeile folgen die Definitionen der Generator-Variablen:

```
Variable1, "Wert der Variable1"
Variable2, "Wert der Variable2"
...
VariableN, "Wert der VariableN"
```

Alle Werte werden in Anführungszeichen eingeschlossen.

Die Textdatei params.txt wird im Fenster zur Einstellung der Umgebung angegeben. Bei einer Datenbank als Datenquelle erstellen wir eine Datenbanktabelle mit zwei Spalten. Die zwei Spalten der Datenbanktabelle werden mit Name und Value benannt. In der Spalte Name werden

◀ **Abbildung 6**
Konfiguration der Datenbank (Eingabe des Pfads und Aliasname der Datenbank)

nacheinander die Variablen als neuer Datensatz eingefügt und in der Spalte Value mit Werten initialisiert.

Die Datenbank mit der einzigen Tabelle »Tabelle1« speichern wir unter dem Namen params.mdb auf der Platte. In diesem Beispiel handelt es sich um eine Microsoft Access-Datenbank.

Jetzt müssen wir die Datenbank params.mdb mit der ODBC-Schnittstelle unter Windows NT für Flash und andere Programme bekannt machen. In der Systemsteuerung (Windows 2000: Systemsteuerung · Verwaltung) befindet sich das Symbol Datenquellen (ODBC).

Unter Benutzer-DSN (für Online-Generierung unter System-DSN) drücken wir die Hinzufügen-Taste und wählen Microsoft Access-Treiber (*.mdb).

Daraufhin erscheint das Fenster ODBC Microsoft Access Setup.

In diesem Fenster drücken wir auf die Auswählen-Taste und geben den Pfad zur params.mdb auf der Platte an. Im Feld Datenquellenname ganz oben im Fenster geben wir den Aliasnamen der Datenbank an. Über diesen Aliasnamen sprechen wir die Datenbank in der Zukunft an. Wir haben als Aliasnamen params angegeben.

Damit Generator die Daten aus der Datenbank params.mdb auslesen kann, geben wir den folgenden SQL-Befehl im Fenster zur Einstellung der Umgebung ein:

```
fgjdbc:///?driver=sun.jdbc.odbc.
 JdbcOdbcDriver&url=jdbc:odbc:params
&userid=&password=&query=SELECT%20*FROM%20Tabelle1;
```

Diese Zeile mag für einige etwas kryptisch und erschreckend aussehen. Diese Zeile erläutern wir etwas näher.

Nur die fett markierten Teile des Befehls müssen von uns geändert werden. `params` ist der Aliasname unserer Datenbank. `SELECT%20*FROM%20Tabelle1`; ist unser SQL-Befehl, mit dem wir alle Datensätze aus Tabelle1 der Datenbank auslesen. Der gesamte Befehl muss ULR-kodiert angegeben werden.

`%20` ist wie gehabt eine Leerzeile in URL-Schreibweise. `userid` und `passwort` müssen nur für geschützte Datenbanken angegeben werden. Für unser Beispiel bleiben die Werte leer.

Somit ist alles getan, damit Generator die Daten aus der Datenbank auslesen kann.

Datenformat für Spaltenname/Spaltenwert

Wenn es sich um eine Textdatei als externe Datenquelle handelt, erstellen wir eine Textdatei params.txt (Name ist beliebig). Die erste Zeile muss alle Variablen wie folgt auflisten:

```
Variable1, Variable2, ... , VariableN
```

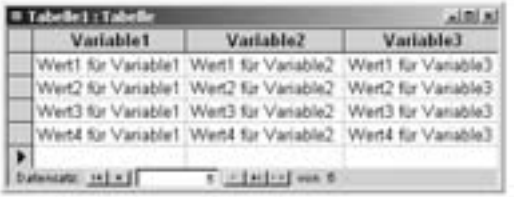
Auf diese Zeile folgen die Zeilen für die Werte der Generator-Variablen:

```
"Wert1 der Variable1", "Wert1 der Variable2", ... , "Wert1 der VariableN"
"Wert2 der Variable1", "Wert2 der Variable2", ... , "Wert2 der VariableN"
...
"WertM der VariableN", "WertM der VariableN", ... , "WertM der VariableN"
```

Alle Werte werden in Anführungszeichen eingeschlossen.

Die Textdatei params.txt geben wir als Datenquelle für Platzhalter (Plot, List etc.) ein.

Bei einer Datenbank als Datenquelle erstellen wir eine Datenbanktabelle mit N Spalten. Die N Spalten der Datenbanktabelle werden mit Variable1 bis VariableN benannt. In der Spalte Variable1 werden nacheinander die Werte für Variable1 eingefügt. Datensatz 1 enthält also den ers-



◀ Abbildung 7
Datenbanktabelle mit Spaltenname/Spaltenwert Format

ten Wert der Variablen 1 bis N, Datensatz 2 enthält den zweiten Wert der Variablen 1 bis N ...

Datensatz M enthält den m-ten Wert der Variablen 1 bis N.

Für den Fall N = 3 und M = 4 zeigen wir eine Tabelle (Abbildung 7).

Damit Generator die Daten aus der Datenbank params.mdb auslesen kann, geben wir den folgenden SQL-Befehl als Datenquelle für Platzhalter (Plot, List etc.) ein:

```
fgjdbc:///?driver=sun.jdbc.odbc.
 JdbcOdbcDriver&url=jdbc:odbc:params
&userid=&password=&query=SELECT%20*FROM%20Tabelle1;
```

Die mehrzeilige Datei im Format Spaltenname/Spaltenwert eignet sich nicht für die Einstellung der Umgebung des Generators. Mit diesem Format weisen wir den verschiedenen Generator-Objekten/-Platzhaltern (Plot, List etc.) Daten zu.

Interne Definition von Generator-Variablen

Im Fenster zur Einstellung der Umgebung können wir, statt eine Datenquelle anzugeben, direkt Umgebungsvariablen definieren.

Den folgenden Text geben wir so im Fenster zur Einstellung der Umgebung ein:

```
#Name, Value
Variable1, "Wert der Variable1"
Variable2, "Wert der Variable2"
```

Damit definieren wir zwei Variablen im Format Name/Wert und weisen ihnen Werte zu.

Noch einmal zum Verständnis:

Bei der Generierung wird jedes Vorkommen von {Variable1} und {Variable2} (siehe »Generator-Variablen definieren« Seite 441) durch die Werte von Variable1 und Variable2 ersetzt.

Benutzung spezieller Zeichen in Datenquellen

Bestimmte Zeichen wie {},\ und " sind für Generator reserviert. Um diese Zeichen in Textdateien benutzen zu können, benutzen wir ein so genanntes Escape-Zeichen.

Für Generator ist dieses Escape-Zeichen der Backslash » \ «.

Ein auf » \ « folgendes Zeichen wird nicht interpretiert, sondern als Teilstring verarbeitet.

- `\r\n` in einem Text verursacht ein Zeilenwechsel. Beispiel: `Variable1, "Zeile1\r\nZeile2"`
- Damit wird {Variable1} nach der Generierung wie folgt dargestellt:
  `Zeile1`
  `Zeile2`
- Mit `Variable1, "Das ist ein \\-Zeichen"` wird {Variable1} wie folgt dargestellt: `Das ist ein \-Zeichen`
- Mit `Variable1, "Das ist ein Zitat: \"Ein großer Schritt für die Menschheit\""` wird {Variable1} nach der Generierung wie folgt dargestellt: `Das ist ein Zitat: "Ein großer Schritt für die Menschheit"`
- In Textdateien dient das Komma zur Trennung der Spalten.
- Wenn Werte in Anführungszeichen eingeschlossen werden, wird das Komma nicht als Trennzeichen interpretiert, sondern als Teilstring des Wertes. Mit `Variable1, "Ein eingeschlossenes Komma bleibt ein/r/n Komma, da es nicht als Trennzeichen interpretiert wird."` wird {Variable1} nach der Generierung wie folgt dargestellt: `Ein eingeschlossenes Komma bleibt ein Komma, da es nicht als Trennzeichen interpretiert wird.`
- Wenn man in Flash ein Textfeld mit {} darstellen will, wird der in {} eingeschlossene Text von Generator als Generatorvariable interpretiert. Um dies zu verhindern und die Klammern als String darzustellen, benutzen wir {{}}. Beispiel: Ein Textfeld soll Folgendes darstellen: `{Text}.` Dafür geben wir {{Text}} in dem Textfeld ein, und nach der Generierung wird in dem Textfeld der String `{Text}` angezeigt und nicht der Wert einer Generatorvariablen.

In Datenbanken muss man keine Escape-Zeichen benutzen (siehe Beispiel zur Scrolling-List Seite 455).

Tutorials zur Einstellung der Umgebung des Generators

Im Verzeichnis ServerScript\Generator\genvariable auf der CD befinden sich drei Beispiele zur Einstellung der Umgebung des Generators. Mithilfe dieser drei Tutorials kann der interessierte Leser leichter nachvollziehen, wie man einem Projekt Umgebungsvariablen im Name/Wert- bzw. Spaltenname/Spaltenformat-Format zur Verfügung stellt.

Nach einem Mausklick auf das Symbol Umgebungsvariablen des Generators wird der Pfad der Datenquelle sichtbar.

Die drei Beispiele sind auf Anhieb nachvollziehbar und brauchen keine gesonderten Erläuterungen.

Tutorials zum Generatorobjekt Table und Insert JPEG File

Hier wollen wir Ihnen zeigen, wie man Tables benutzt und Eigenschaften von Objekten als Variablen angibt.

Im Verzeichnis Genenerator\table befindet sich ein Beispiel zur Erstellung einer Tabelle. In dieser Tabelle sollen Bilder angezeigt werden. Die Namen der Bilder und die JPEG-Qualität sollen in weiteren Spalten der Tabelle angezeigt werden. Einige Eigenschaften der Tabelle im Generatorinspektor werden durch Generatorvariablen angegeben. Damit wird nicht nur der Inhalt der Tabelle variabel, sondern auch dessen Eigenschaften. Diese Eigenschaftsvariablen beziehen wir aus der Textdatei tab_params.txt. tab_params.txt geben wir als Datenquelle im Fenster zur Einstellung der Umgebung ein.

tab_params.txt ist wie folgt aufgebaut:

```
Name, Value
quelldatei, "table.txt"
rows, "3"
cols, "3"
RowLabel, "Bild1,Bild2,Bild3"
ColLabel, "Bild,Name,Qualität"
```

Die Variable `quelldatei` mit dem Wert `table.txt` wird als »Data Source«-Eigenschaft der Tabelle angegeben. Während der Generierung bezieht die Tabelle ihre Eigenschaften aus der Textdatei table.txt. Die Variablen `row` und

`cols` mit den Werten 3 und 3 werden als Rows und Columns-Eigenschaften der Tabelle angegeben. Somit ist die Größe der Tabelle auf 3 x 3 festgesetzt.

Die Variablen `RowLabel` und `ColLabel` sind die Beschriftungen der Zeilen und Spalten der Tabelle. Diese Variablen weisen wir den Eigenschaften `Row Labels` und `Column Labels` der Tabelle zu. Lässt man diese Eigenschaften leer, sind Zeilen und Spalten nicht beschriftet. Die Beschriftungen werden durch Kommata getrennt, die Spalten der Tabelle sind in diesem Beispiel mit Bild, Name und Qualität benannt.

Da wir die Eigenschaften als Variable definiert haben, ist es uns möglich, mit der Änderung der Textdatei tab_params.txt die Eigenschaften der Tabelle zu ändern.

Wir benötigen zwei Symbole, mit denen wir die Tabelle füllen:

Der erste Symbol ist ein MovieClip mit einer Generatorvariablen `var1` auf der Bühne. Der zweite MovieClip beinhalten einen INSERT JPEG FILE-Platzhalter. Als File Name-Eigenschaft des Bildes geben wir `var1` ein und als Quality-Eigenschaft geben wir `var2` ein. `Scale To Fit` wird auf `True` gesetzt. Dadurch wird das Bild so verändert, dass es den Platzhalter füllt. Die Symbole werden in der Bibliothek im Ordner style angelegt und mit `text` und `bild` benannt.

Als Quelle hatten wir table.txt angegeben. In table.txt geben wir den Inhalt der Tabelle an. Die Table-Quelldatei benötigt zumindest die Spalte Clip in der ersten Zeile. In den darauffolgenden Zeilen werden die Daten blockweise angegeben. Der erste Block definiert den Inhalt der ersten Zeile der Tabelle, der zweite Block definiert den Inhalt der zweiten Zeile der Tabelle usw.

table.txt ist wie folgt aufgebaut:

```
Clip, var1, var2

style/bild, "001.jpg", "1"
style/text, "Die Dateiname\rlautet\r001.jpg",
style/text, "1%",

style/bild, "002.jpg", "50"
style/text, "Die Dateiname\rlautet\r002.jpg",
style/text, "50%",

style/bild, "003.jpg", "100"
style/text, "Die Dateiname\rlautet\r003.jpg",
style/text, "100%",
```

In der ersten Zeile definieren wir die benötigten Spalten/Variablen.

Die Spalte Clip ist zwingend erforderlich. Mit dieser Spalte legen wir fest, welches Symbol aus der Flashbibliothek eingefügt werden soll.

Die Datei hat drei Blocks mit Werten. Jeder Block definiert den Inhalt einer Zeile der Tabelle. Somit wird die ersten Zeile der Tabelle wie folgt aufgebaut:

Die erste Spalte der Zeile 1 beinhaltet das Symbol `bild`. Das Bild 001.jpg wird der Variablen `var1` zugewiesen und der INSERT JPEG FILE-Platzhalter benutzt diese Variable als Source File-Eigenschaft. Der Wert 1 wird der Variablen `var2` zugewiesen und der INSERT JPEG FILE-Platzhalter benutzt diese Variable als Quality-Eigenschaft.

Somit wird in der ersten Spalte der Zeile 1 das Bild 001.jpg mit einer Qualität von 1% angezeigt.

Die zweite Spalte der ersten Zeile soll den Dateinamen und die dritte Spalte soll die Qualität in Prozent anzeigen. Diese Werte werden der Variablen `var1` in dem Symbol `text` zugewiesen. `var2` bleibt für die zweite und dritte Spalte leer, da das Symbol `text` nur `var1` benötigt. Die zweite und dritte Zeile der Tabelle werden wie die erste Zeile in jeweils einem neuen Block definiert.

Wenn wir die Tabelle erweitern möchten, fügen wir einen neuen Block hinzu und erhöhen den Wert für `rows` in der Textdatei tab_params.txt. Die Erweiterung der Tabelle ist möglich, da wir die Eigenschaften der Tabelle als Variablen angegeben hatten.

Tutorial zum Generatorobjekt Plot

Hier zeigen wir Ihnen, wie man mit Hilfe von `Plot` Symbole frei positionieren kann. Wir werden ein dynamisches Menü basteln.

Das Plot-Objekt ist wohl das mächtigste Generatorobjekt überhaupt. Es ist leicht vorstellbar, eine komplette Website mit Hilfe des Plot-Objekts zu erstellen. `Plot` bietet die Möglichkeit, beliebig viele verschiedene Symbole frei zu positionieren und frei zu transformieren.

Im Verzeichnis Genenerator\menue befindet sich ein Beispiel zur Erstellung eines dynamisch erzeugten Menüs. Wir haben vor, eine Drop-down-Menüstruktur aufzubauen. Jedes Drop-down-Menü soll höchstens fünf Unterpunkte besitzen.

Wir erstellen hierfür ein Tastensymbol. Dieses Tastensymbol besitzt in den vier Statusframes jeweils ein Rechteck. Die Taste positionieren wir im ersten Frame des `MenüBtnMC`-MovieClips. Den Clip `MenüBtnMC` erweitern

wir mit fünf weiteren Frames. In allen Frames ist die obige Haupttaste sichtbar.

Über dieser Taste positionieren wir ein statisches Textfeld mit der Generatorvariablen `menuepunkt` als Inhalt. Diese Variable soll nach der Generierung den Titel der Taste anzeigen. Eine Ebene tiefer positionieren wir im zweiten Frame eine weitere Instanz des Tastensymbols. Diese Instanz dient als erster Untermenüpunkt. Wir positionieren sie direkt unterhalb der Haupttaste und modifizieren die Instanz ein wenig, so dass sie sich im Aussehen von der Haupttaste unterscheidet. Wir positionieren über dieser Taste ein statisches Textfeld. Das Feld beinhaltet die Generatorvariable `punkt1`.

Eine weitere Ebene tiefer positionieren wir den zweiten Untermenüpunkt und verfahren wie beim ersten. Das Ganze wiederholen wir für die restlichen drei Untermenüpunkte.

Zum Schluss haben wir in jeder Ebene eine weitere Taste erstellt. Im ersten Frame ist nur die Haupttaste sichtbar, im zweiten Frame die Haupttaste und der erste Untermenüpunkt, im dritten Frame die Haupttaste und die ersten beiden Untermenüpunkte usw. Das zweite Frame wird mit `_1` benannt, die dritte mit `_2` und schließlich wird das sechste Frame mit `_5` benannt. Die Bildnamen leitet sich aus der Anzahl der sichtbaren Untermenüpunkte ab.

Wir erstellen eine neue Ebene an der untersten Stelle. In diese Ebene positionieren wir im zweiten Frame eine sehr große unsichtbare Taste (die Funktion wird unten erläutert).

Die Haupttaste besitzt als Action:

```
on (rollOver) {
   gotoAndStop ("{anzahl}");
}
```

`anzahl` ist eine Generatorvariable, die festlegt, wie viele Untermenüpunkte dargestellt werden sollen. `anzahl` nimmt die Werte `_1` bis `_5` an. Wenn man mit der Maus über die Haupttaste rollt, wird in das Frame gesprungen, das durch `anzahl` angegeben wird.

Die große, unsichtbare Taste hat als Action:

```
on (rollOver) {
   gotoAndStop(1);
}
```

Wenn das Menü aufgeklappt ist (ab Frame 2), soll es wieder geschlossen werden, wenn man mit der Maus über die Taste rollt. Die Untermenütasten besitzen als Action:

```
on (release) {
   _root.ZielClip.gotoAndStop("{zielframe1}");
}
```

Wobei das zweite Untermenütaste **zielframe2** als Sprungziel besitzt usw. **ZielClip** ist ein MovieClip auf der Bühne. In diesem Clip wird visualisiert, welche Taste als letztes gedrückt wurde.

Jetzt kommen wir zum interessanten Teil.

Wir positionieren ein Plot-Objekt auf der Bühne und geben als Data Source-Eigenschaft **plot_menuestruktur.txt** ein. Plot bezieht jetzt seine Daten aus dieser Textdatei.

plot_menuestruktur.txt ist wie folgt aufgebaut:

```
Clip, X, Y, menuepunkt, anzahl, punkt1, zielframe1, punkt2,
zielframe2, punkt3, zielframe3, punkt4, zielframe4, punkt5,
zielframe5
```

Die erste Zeile listet alle benötigten Spalten und Variablen auf, wobei bei jedem Plot-Objekt die Spalten Clip, X und Y zwingend erforderlich sind. Clip bezeichnet das Symbol, das eingefügt werden soll. X und Y definieren die Position des einzufügenden Symbols, wobei (0,0) die obere, linke Ecke der Plot-Objekts bezeichnet.

menüpunkt ist der Titel der Haupttaste und mit **anzahl** legen wir fest, wie viele Untermenüpunkte wir dieser Haupttaste zuweisen wollen. **punkt1** ist der Name des ersten Untermenüpunktes und **zielframe1** ist die Sprungmarke, die dem ersten Untermenüpunkt zugewiesen wird. **punkt2**, **zielframe2** bis **punkt5**, **zielframe5** sind die Wertepaare der restlichen vier Untermenüpunkte.

```
Buttons/MenüBtnMC, 10, 20, MENÜPUNKT1, _2, PUNKT1_1, mp1_1,
PUNKT1_2, mp1_2, , , , , ,
```

Mit dieser Zeile positionieren wir das erste Drop-down-Menü im Plot-Objekt. Das Menü wird an der Position (10, 20) eingefügt und die Haupttaste bekommt als Titel **menuepunkt** den Wert **MENÜPUNKT1** zugewiesen.

Die Taste soll zwei Untermenüpunkte besitzen. Wir weisen daher `anzahl` den Wert `_2` zu. Bei einem Rollover über die Haupttaste wird zum Frame mit dem Namen `_2` gesprungen. In diesem Frame sind, wie oben beschrieben, die ersten zwei Untermenüpunkte sichtbar.

Der Untermenüpunkt bekommt als Titel `punkt1` zugewiesen und als Wert `PUNKT1_1`. Bei einem Klick auf diese Taste soll im ZielClip zum Frame mit der Bezeichnung `{zielframe1} mp1_1` gesprungen werden.

Für den zweiten Untermenüpunkt verfahren wir wie für den ersten.

Da nur zwei Untermenüpunkte sichtbar sind, weisen wir den restlichen Variablen `punkt3`, `zielframe3` bis `punkt5`, `zielframe5` keine Werte zu. Ein Komma signalisiert Generator, dass der Wert für die nächste Spalte beginnt. Gibt man nach dem Komma keinen Wert ein, bleibt der Wert der Variablen leer. Die Kommata sind zwingend erforderlich.

Zum besseren Verständnis: Besitzt eine Textdatei die Spalten `Clip, var1, var2,` und `var1` soll kein Wert zugewiesen werden, dann muss die Datenzeile wie folgt aussehen: `ClipName,, "Wert für var2".`

Das zweite Komma signalisiert Generator, dass jetzt der Wert für `var2` angegeben wird und nicht der von `var1`.

```
Buttons/MenüBtnMC, 40, 20, MENÜPUNKT2, _5, PUNKT2_1, mp2_1,
PUNKT2_2, mp2_2, PUNKT2_3, mp2_3, PUNKT2_4, mp2_4, PUNKT2_5, mp2_5

Buttons/MenüBtnMC, 70, 20, MENÜPUNKT3, _3, PUNKT3_1, mp3_1,
PUNKT3_2, mp3_2, PUNKT3_3, mp3_3, , , ,
```

Mit diesen beiden Zeilen fügen wir zwei weitere Haupttasten in das Plot-Objekt ein. Die zweite Taste hat fünf Untermenüpunkte und die dritte drei. Jetzt haben wir eine Menüstruktur auf der Bühne. Jede Haupttaste hat eine andere Anzahl an Untermenüpunkten und führt eine andere Aktion durch.

Man kann dieses Beispiel so erweitern, dass die Anzahl der Untermenüpunkte nicht auf fünf begrenzt ist. Das erreichen wir, indem wir im `MenüBtnMC`-MovieClip den Befehl `myMC.duplicateMovieClip( newName, depth )` benutzen, um neue Untermenüpunkte zu erstellen.

Das Plot-Objekt besitzt weitere Eigenschaften, die wir in diesem Tutorial nicht benutzt haben. Die Quelldatei kann weitere optionale Spalten beinhalten:

Die Spalten X-Scale und Y-Scale geben die Skalierung des einzufügenden Symbols an. Mit der Spalte Rotate wird das einzufügende Symbol um den Wert von `Rotate` rotiert. Gibt man diese Spalten in der Quelldatei

nicht an, werden die Originalwerte des Symbols benutzt. Diese optionalen Spalten eröffnen uns weitere unerschöpfliche Möglichkeiten.

Tutorial zum Generatorobjekt Scrolling List

Hier zeigen wir Ihnen, wie man mit Hilfe von Scrolling List die Texte einer Website dynamisch erzeugen kann. Die Texte werden in verschiedenen Blöcken angezeigt, wobei jeder Block andere Eigenschaften besitzt. Mit Eigenschaften meinen wir die Darstellungsform eines Textblocks. Diese umfasst die Schriftart, Größe, Farbe, Ausrichtung des Textes usw. Des Weiteren weisen wir bestimmten Textblöcken eine Funktion zu, so kann z. B. mit einem Klick auf eine Zeile zu einer URL gesprungen werden, ein Mail versendet oder ein Hintfeld angezeigt werden. Und das alles wird dynamisch zugewiesen. In einer Scrolling List können, genau wie im Table-Tutorial gezeigt, weitere Platzhalter eingefügt werden. Wir werden unserer Scrollig List Bilder zwischen Textblöcken hinzufügen.

Wir behandeln hier zwei verschiedene Datenquellen.

1. Das Objekt `Scrolling List` bezieht seine Daten aus Textdateien.
2. Das Objekt `Scrolling List` bezieht seine Daten aus Datenbanktabellen.

Die Firma WebVision2000 (www.webvision2000.de) hat ihre Seiten mit der Datenbankvariante erstellt und stellt uns diese Technik zur Verfügung.

Im Verzeichnis Genenerator\scrolllist finden sich zwei Beispiele zur Erstellung dynamischer Textblöcke. scrolllist-db.fla ist die Datenbankvariante und scrolllist-txt.fla bezeichnet die Textdateivariante.

Mithilfe eines Vergleich der zwei Datenquellen (Datenbank und Textdatei) dürfte dem Leser schnell bewusst werden, welche Merkmale bei der Erstellung der verschiedenen Datenquellen zu beachten sind.

Symbole erstellen

Wir beginnen mit der Erstellung der benötigten Symbole. Diese Symbole werden anschließend in die Scrollig List eingefügt. Zunächst erstellen wir zehn MovieClips. Jeder MovieClip beinhaltet ein statisches Textfeld mit der Generatorvariablen `text` als Inhalt. Die Textfelder erhalten unterschiedliche Textgrößen, Farben etc.

Wir benennen die Symbole mit a0 bis a9 und verschieben sie in den Ordner style der Flashbibliothek.

Dann erstellen wir einen neuen MovieClip. Dieser beinhaltet ein Textfeld mit der Generatorvariable **hint** und einen braunen Hintergrund. Außerdem benötigen wir zwei weitere MovieClips. Mithilfe des ersten soll es möglich sein, eine URL aufzurufen, mit der zweiten soll man eine Mail versenden können.

Wir nennen diese beiden Clips **link** und **mail**. Sie befinden sich ebenfalls im Ordner style. Der Clip **link** beinhaltet neben einem Textfeld mit der Generatorvariablen **text** eine dreieckige Taste und eine Instanz unseres Clips **hint**. Der Clip **hint** wird im ersten Frame unsichtbar gemacht. Die Taste im Clip **link** hat als Action:

```
on (release) {
   setProperty ("hint", _visible, false);
   stopDrag ();
   getURL ("{link}", "_blank");
}
on (rollOver) {
   startDrag ("hint", true);
   setProperty ("hint", _visible, true);
}
on (releaseOutside, rollOut) {
   setProperty ("hint", _visible, false);
   stopDrag ();
}
```

Bei einem Roll-over wird der Clip **hint** sichtbar und verfolgt die Maus. Bei einem Roll-out verschwindet er wieder. Bei Betätigung der Taste wird die URL **link** in einem neuen Fenster aufgerufen.

Es ist möglich, die Eigenschaft **_blank** ebenfalls durch eine Variable anzugeben. Dies geschieht z..B mit: **getURL ("{link}", "{form}");**

Hat **form** als Wert **_blank,** wird ein neues Fenster geöffnet. Ist der Wert **_self**, wird die URL im selben Fenster geöffnet usw... Der Clip **mail** ist wie der Clip **link** aufgebaut. Die Taste im Clip **mail** hat als Aktion die selben Aktionen wie die Taste **link**. Statt **_blank** ist die Eigenschaft im Clip **mail** auf **_self** gesetzt.

Wir erstellen einen MovieClip mit einem INSERT JPEG FILE-Platzhalter als Inhalt. Die Eigenschaft **File Name** des INSERT JPEG FILE-Platzhalters wird auf die Variable **text** gesetzt. Dieser Clip wird **pic** genannt. Wir er-

stellen einen neuen MovieClip mit dem Symbolnamen `bild`. Dieser Clip beinhaltet mehrere Instanzen des Clips `pic`. Jede Instanz ist verschieden transformiert (Transformation der Größe und Farbwerte).

Wir positionieren die Instanzen in einem Bereich auf der Bühne, der fast genauso breit ist wie die Scrolling List in der Hauptzeitleiste. Wir positionieren eine Instanz des Clips `hint` und eine große, unsichtbare Taste verdeckt alle Instanzen im Clip `bild`. Mit dieser Taste wird wie im Clip `link` der Clip `hint` an- und ausgeschaltet.

Jetzt stehen uns alle benötigten MovieClips zur Verfügung.

Wir positionieren im ersten Frame der Hauptzeitleiste eine Instanz vom Scrolling List. Für die Eigenschaft `Orientation` stellen wir `vertical` ein, damit werden die Elemente untereinander positioniert und die Liste vertikal gescrollt. Mit der Eigenschaft `Step Size` stellen wir fest, wie viele Pixel bei jedem Schritt gescrollt werden soll.

Zum ersten Mal benutzen wir die wichtige Eigenschaft `Instance name` eines Generatorobjekts. Als `Instance name` geben wir `List` ein. Damit kann ein Platzhalter in ActionScripts abgesprochen werden.

Als `Data Source` geben wir die Textdatei Willkommen.txt ein. Im zweiten Frame positionieren wir eine Kopie unseres Scrolling List-Objekts. Die Eigenschaft `Instance Name` wird nicht verändert. Wir verändern lediglich die Eigenschaft `Data Source`. Als Wert für `Data Source` unseres zweiten Scrolling List-Objekts geben wir Bilder.txt ein. Ist die Eigenschaft `Mask To Box` auf `true` gesetzt, wird der sichtbare Bereich durch die Begrenzungsrahmen des Scrolling List-Objekts begrenzt.

Wir positionieren vier Tasten auf der Bühne. Zwei Tasten dienen zum Scrollen der Listen und mit den anderen beiden Tasten springen wir in die zwei Frames der Haupzeitleiste. Im ersten Frame ist die erste Scrolling List mit dem Inhalt von Willkommen.txt sichtbar und im zweiten Frame sehen wir die zweite Scrolling List mit dem Inhalt von Bilder.txt.

Die Scroll-Tasten haben als Actions:

```
on (press) {
   tellTarget ("List") {
      prevFrame ();
   }
}
```

Mit dieser Taste wird nach oben gescrollt und mit der zweiten nach unten.

```
on (press) {
   tellTarget ("List") {
      nextFrame ();
   }
}
```

Die Textdatei Willkommen.txt ist wie folgt aufgebaut:

Die erste Zeile definiert wieder in Spaltenname/Spaltenwertformat die Spalten und Variablen der Quelldatei. Die Spalte Clip ist zwingend erforderlich.

```
Clip, text, link, hint
```

Es folgt eine Leerzeile. Für das weitere Vorgehen siehe die Erläuterungen unter Willkomen.txt

Der Text in Willkommen.txt erläutert die Benutzung der obigen MovieClips im Ordner style der Flashbibliothek.

```
style/a0, , ,
style/a2, "ERLÄUTERUNGEN", ,
Die Textdatei Bilder.txt ist wie folgt aufgebaut:
Clip, text, link, hint
style/a0, , ,
style/a2, "BILDER", ,
style/a0, , ,
style/a3, "Hier zeigen wir euch wie man den Style style/bild aus
unserer Flash Bibliothek benutzen kann.", ,
style/a0, , ,
style/bild, "001.jpg", "mailto:you@somewhere.de", "mailto:you@somew-
here.de"
```

usw.

Als erstes Bild wird 001.jpg eingefügt. Wir hatten die Variable `text` als Eigenschaft `File Name` des INSERT JPEG FILE-Platzhalters angegeben, somit wird in der Spalte text die Bilddatei angegeben, die angezeigt werden soll.

Bei einem Roll-over erscheint der Text in der hint-Spalte. Bei einem Roll-out verschwindet er wieder. Die Spalte link beinhalten die URL-kodierte Mailadresse.

Die Datenbanktabellen »Willkommen« und »Bilder« in der Datenbank content.mdb sind ähnlich wie die Textdateien aufgebaut.

Um die Daten aus der Datenbank zu beziehen, geben wir als »Data Source« für die erste Liste den folgenden Befehl ein:

```
fgjdbc:///?driver=sun.jdbc.odbc.JdbcOdbcDriver&url=jdbc:odbc:content
&userid=&password=
&query=SELECT%20*%20FROM%20Willkommen%20ORDER%20BY%20context;
```

`content` ist die ODBC-Aliasname der Datenbank content.mdb (siehe Umgebungsvariablen und Datenformate). Die `Select`-Anweisung ist die URL-kodierte SQL-Abfrage der Daten aus der Tabelle »Willkommen«. Die Datensätze werden mit Hilfe der Spalte »context« sortiert. Im Text der Tabelle »Willkommen« wird die Funktion von »context« erläutert. (Starten Sie dazu die Datei scrolllist-db.swf.)

Wir geben als `Data Source` für die zweite Liste den folgenden Befehl ein:

```
fgjdbc:///?driver=sun.jdbc.odbc.JdbcOdbcDriver&url=jdbc:odbc:content
&userid=&password=
&query=SELECT%20*%20FROM%20Bilder%20ORDER%20BY%20context;
```

Damit ist dieses Tutorial abgeschlossen.

Weitere Fähigkeiten von Generator

Das ganze Potenzial von Generator ist mit diesem Kapitel längst nicht erschöpft. Wir haben viele der Generatorobjekte nicht im Detail durchgearbeitet. Dies hätte den Rahmen des Kapitels gesprengt. Wir haben jedoch alles Wichtige im Detail behandelt. Der interessierte Leser kann die weiteren Generatorobjekte mit Hilfe dieses Kapitels leicht nachvollziehen.

Kurze Bemerkungen zu den weiteren Objekten von Generator

- `Insert Symbol`: Mit diesem Objekt können wir ein beliebiges Symbol aus der Bibliothek einfügen.
- `Insert Flash Movie`: Mit diesem Objekt können wir dynamisch Movies in das SWF-Format einfügen.

- `Insert Sound`, `Insert MP3 File`: Mit diesen Objekten können wir dynamisch Sounds einfügen.
- `List`, `Ticker` und `Multipage List`: Diese Objekte ähneln dem `Scrolling List`-Objekt. Unterschiede: `List` kann nicht gescrollt werden, `Ticker` scrollt automatisch in einer Endlosschleife und `Multipage List` bekommt drei Tastensymbole zugewiesen; mit diesen Tasten kann man Seiten durchblättern, z. B. zur Anzeige von Adressen aus einer Datenbank.
- `Basic Charts`, `Pie Chart` und `Stock Chart`: Diese Objekte machen Generator einmalig. Mit diesen Objekten können Daten aus einer Datenbank in Form von zwölf verschiedenen Charttypen visualisiert werden. Diese Objekte eignen sich hervorragend für Statistiken, der Darstellung von Börsenkursen, E-Commerce-Anwendungen etc.
- `Radio Button`: Mit diesem Objekt werden dynamische Radio-Buttons generiert.

Kurze Bemerkungen zu den weiteren Fähigkeiten von Generator

Generator bietet die Möglichkeit, Symbole auf der Bühne dynamisch zu verändern. Wenn man ein Symbol auf der Bühne auswählt, werden die folgenden Generatorkommandos im Generatorinspektor aktiviert:

- Als Standard ist jedem Symbol auf der Bühne `Keine Auswahl` zugeordnet. Damit bleibt ein Symbol statisch wie auch unter Flash.
- Das Kommando `Replace` ersetzt ein Symbol auf der Bühne durch ein anderes aus der Bibliothek.
- `Replicate` spielt ein Symbol in einer Sequenz ab. Jede Sequenz (komplettes Abspielen eines Clips bis zu seinem Ende) stellt die Daten der nächsten Zeile der zugewiesenen Datenquelle dar.
- Mit `Set Alpha`, `Set Brightness`, `Set Custom Color` und `Set Tint` können die Eigenschaften eines Symbols dynamisch verändert werden.
- Mit `Transform` können Symbole auf der Bühne repositioniert und transformiert werden. Dabei kann ein Symbol um einen Offset-Wert verschoben, gedreht und/oder beliebig vergrößert oder verkleinert werden.

Swift-Generator

Seit langem wird ein Tutorial zum Swift-Generator vermisst, da es zu diesem Programm bis jetzt nur englische und französische Anleitungen gibt und in den entsprechenden Newsgroups immer wieder nach deutschsprachigen Informationen gefragt wird.

▶ Stephan Fischer
www.phaetons.de

Zunächst erhalten Sie eine kurze Übersicht, was man mit Swift-Generator überhaupt machen kann und wo der Unterschied zu dem Macromedia Generator liegt. Dann gibt es eine allgemeine Übersicht über die Funktionsweise und die einzelnen Arbeitsschritte. Verweise in der Übersicht ermöglichen es, auch direkt zu den Details zu springen, so dass man sowohl Schritt für Schritt als auch themenorientiert vorgehen kann.

Einige wenige Absätze habe ich (mit freundlicher Genehmigung von Olivier Debon) direkt aus der englischen Anleitung übernommen, da es keinen Sinn gehabt hätte, hier im Wortlaut irgendetwas zu ändern.

Was ist der Swift-Generator?

Swift-Generator wurde von Olivier Debon entwickelt und kann dazu benutzt werden, online oder offline Flash-Dateien dynamisch zu erzeugen.

Möchte man Inhalte oder Parameter einer SWF-Datei ändern, muss man normalerweise die original Flash-Datei öffnen, die neuen Inhalte (wie z. B. Bilder oder Sounddateien) einfügen oder Parameter ändern und dann eine neue SWF-Datei erzeugen, die dann online gestellt wird.

Swift-Generator ermöglicht es, mittels einer Skriptdatei die neuen Inhalte in eine Flash-Vorlage einzulesen und das Ergebnis als SWF-Datei auszugeben. Wie das genau funktioniert, erläutere ich später.

Man muss also die original Flash-Datei nicht mehr verändern, sondern kann neue oder personalisierte Inhalte in Echtzeit erzeugen. Welche Zeitersparnis das bringt, brauche ich hier wohl nicht weiter zu erklären.

Swift-Generator arbeitet mit Flash-Dateien ab Version 3 und kann folgende Objekte beeinflussen:

- Text (Größe, Ausrichtung, Kerning, Farbe,Transparenz und Schriftart)
- Textfelder (ab Flash 4)
- Fonts
- Sounddateien
- Bilddateien (JPEG, PNG)
- Flash-Filmsequenzen
- ActionScript-Parameter (ab Flash 4, in Schlüsselbildern und Buttons)

Macromedia Generator oder Swift-Generator ?

Jetzt mögen einige denken, das kann Macromedia Generator ja auch, warum sollte ich Swift-Generator benutzen? Beide Werkzeuge haben Vor- und Nachteile, über die man sich im Klaren sein sollte, bevor man sich für eines entscheidet. Ich möchte hier ein paar Argumente anführen, welche die Entscheidung erleichtern sollten (diese haben keinen Anspruch auf Vollständigkeit):

Einer der größten Unterschiede ist wohl der Preis. Macromedia bietet den Generator zurzeit in zwei Versionen an. Zum einen gibt es die Enterprise Edition ab US-Dollar 30.000. Diese Version ist für große Websites gedacht, in denen dynamische Flash-Daten in großer Zahl und in Echtzeit generiert werden sollen. Diese Version ist über mehrere CPUs und Server skalierbar, und Macromedia bietet einen 24-Stunden-Support. Außerdem speichert die Enterprise Edition Daten in einem Cache und kann somit bei Bedarf schneller auf diese Daten zugreifen.

Die zweite Version nennt sich Developer Edition und ist für US-Dollar 999 pro Server erhältlich. Sie richtet sich an Entwickler, die Flash-Dateien regelmäßig updaten oder personalisieren müssen, aber auf Online-Erzeugung verzichten können. Die Developer Edition kann im Gegensatz zur Enterprise Edition nur einen Prozeß nach dem anderen bearbeiten, ist nicht skalierbar und erzeugt keine Cachedaten, hat somit also eine geringere Performance. Der Support beschränkt sich auf 90 Tage.

Swift-Generator ist Freeware unter der Voraussetzung, dass man das Swift-Logo auf seiner Website verlinkt, oder man kann sich gegen eine Gebühr von 100 € pro Domain registrieren lassen, ohne das Swift-Logo benutzen zu müssen. Support ist nicht garantiert, Anfragen meinerseits wurden aber grundsätzlich innerhalb 48 Stunden beantwortet. Swift-Generator erzeugt keine Cachedaten, die Performance bei Online-Betrieb kann sich aber durchaus sehen lassen.

Macromedia Generator fügt sich in die Oberfläche von Flash mit ein, während Swift-Generator ein skriptgesteuertes Programm ist, also um einiges unkomfortabler zu handhaben ist.

Macromedia Generator läuft auf Windows (95/98/NT/2000), Linux Red Hat und Solaris (ab 2.6). Von Swift-Generator sind Binarys erhältlich für alle Linux-Plattformen (x86/Alpha/MiPS/PPC), Windows (95/98/NT/2000), Solaris (ab 2.6), FreeBSD, BSDi und seit kurzem auch für Mac OS X.

Macromedia Generator kann auf diverse Datenbanken zugreifen, Swift-Generator kann seit Version 0.15 auf ODBC-/MySQL-Datenbanken zugreifen.

Seit Version 0.20.1 unterstützt Swift-Generator auch Bilddateien im Format GIF.

Sowohl Macromedia Generator als auch Swift-Generator sind inzwischen vollständig kompatibel mit Flash 5.

Generell kann man sagen, der MM-Generator ist komfortabler und bietet umfangreichere Funktionen, ist aber auch entsprechend teuer. Wer das Scripting nicht scheut und auch nicht so ein großes Budget zur Verfügung hat, kommt mit dem Swift-Generator in vielen Fällen ans gleiche Ziel.

Weitere Hinweise

Unter www.swift-tools.com/swift-users.html findet man eine inzwischen umfangreiche Liste von Firmen, die mit Swift-Generator arbeiten, unter anderem Bayern 3 Radio, Toyota Motorsports, Coca Cola und startmovie.net ....

Deutschsprachige Informationen zum Programm sind selten zu finden, unter www.flashkit.com/board/index.php findet man ein englischsprachiges Swift-Generator-Forum, in dem man auch auf deutschsprachige Anwender trifft.

Übersicht über die Funktionsweise

Die Vorgehensweise, um mit Swift-Generator eine Flash-Datei zu erzeugen, kann in vier einzelne Schritte eingeteilt werden:

1. Erstellen einer Vorlage (Template oder Flash-Datei)
2. Einbinden eventueller Swift-Variablen in die Vorlage
3. Erstellen der Swift-Skriptdatei

4. Erzeugen der dynamischen Flash-Datei per Kommandozeilenaufruf (offline) oder CGI (online)

Erstellen der Vorlage – Templates oder Flash-Datei?

Als Vorlage für Swift-Generator kann man Template-Dateien (*.swt) oder normale Flash-Dateien (*.swf) benutzen. Beide Dateiarten können mit der Flash-Software erzeugt werden. Eine Template-Datei enthält wesentlich mehr Informationen (so zum Beispiel Schriftartinformationen etc.) als eine Standard-Flash-Datei. In einem Template sind alle nötigen Informationen vorhanden, um z. B. Text auszutauschen, während bei einer normalen Flash-Datei auf jeden Fall die TrueType-Schrift installiert und entsprechende Verweise im Swift-Skript stehen müßten.

Template-Dateien sind also vorzuziehen, da einfacher zu handhaben. Flash-Dateien benutzt man eigentlich nur, wenn diese z. B. von einer anderen Agentur geliefert wurden und man keinen Zugriff auf die Originaldateien hat.

Hier gehe ich auf die Vorgehensweise und Befehle im Einzelnen ein. Unter http://www.derijen.nl/Peerless/ finden Windows-Benutzer ein kostenloses Frontend namens SGui. Diese visuelle Oberfläche erleichtert die Erstellung des Skripts wesentlich. Man kann aber auch jeden Texteditor wie Notepad oder einen der vielen Freeware-Editoren benutzen.

Die Vorlage erzeugen

Als Grundgerüst für die dynamisch zu erzeugende Flash-Datei benötigt Swift-Generator eine Vorlage, die zum einen statische Elemente enthält, zum anderen Variablen als Platzhalter für die dynamisch einzusetzenden Inhalte besitzen kann. Diese Vorlage kann entweder eine normale Flash-Datei (*.swf) sein oder eine mit der Exportfunktion von Flash erzeugte Template-Datei (*.swt). Eine Template-Datei ist in der Regel vorzuziehen, da sie im Gegensatz zur normalen Flash-Datei mehr Informationen enthält, insbesondere die Schriftartdefinitionen.

Um eine Template-Datei zu erstellen, öffnet man die Arbeitsdatei (*.fla) und wählt dann Datei · Film exportieren…

Im folgenden Speicherdialog wählt man dann als Dateityp Generator Vorlage (*.swt).

Abbildung 1 ▶
Als Generator-Vorlage exportieren

Abbildung 2 ▶
In Textelementen können Variablen direkt eingefügt werden

Swift-Variablen in die Vorlage einfügen

Um Text- oder ActionScript-Parameter in der Flash-Vorlage auszutauschen, werden an den entsprechenden Stellen Variablen eingefügt, die Swift-Generator anhand eines Skriptes mit den neuen Inhalten ersetzt.

Andere Elemente (Images, Sound, Filmsequenzen) wiederum benötigen keine Variablen als Platzhalter, da sie über ihre Symbol- bzw. Instanznamen oder ID identifiziert werden können.

Das Wichtigste an der Vorlage sind die einzufügenden Swift-Variablen, die als Platzhalter für die dynamisch einzufügenden Texte oder Parameter dienen. Über das später erstellte Swift-Skript erfährt Swift-Generator genau, welche Variablen durch welche Inhalte auszutauschen sind.

Diese Swift-Variablen sind nicht zu verwechseln mit den normalen Flash-Variablen und werden definiert, indem man sie in geschweifte Klammern setzt:

```
{Swift-Variable}
```

◀ Abbildung 3
LoadMovie mit einer Swift-Variablen

Variablen in Textelementen

Die Variable {name} kann man jetzt ersetzen, indem man im Swift-Skript Folgendes angibt:

```
SET name "Welt"
```

In der mit Swift-Generator erzeugten Flash-Datei stünde dann:

```
Hallo Welt
```

Achtung: Bei mehrzeiligen Texten würde diese Methode nur die erste Zeile an Stelle der Variablen einfügen! Um mehrzeilige Texte zu ersetzen, gibt es die SUBSTITUTE TEXT-Anweisung.

Variablen in ActionScript-Parametern

Swift-Variablen können auch in ActionScript-Parametern eingebunden werden, z. B. der LoadMovie-Befehl, den Sie in Abbildung 3 sehen.

Daraus wird über die Skript-Anweisung

```
SET moviename "movie2.swf"
```

nach Erzeugung des neuen Filmes mittels Swift-Generator

```
LoadMovie ("movie2.swf",1)
```

In welchen Parametern von ActionScript-Befehlen kann man Swift-Variablen einsetzen?

Aktion	Parameter
`GetUrl`	▶ URL ▶ Fenster
`GoTo`	▶ Label
`FS Command`	▶ Befehl ▶ Argumente
`LoadMovie`	▶ URL ▶ Ziel
`UnloadMovie`	▶ Ebene
`TellTarget`	▶ Ziel
`SetVariable`	▶ Variable ▶ Wert

Das Swift-Skript

Swift-Skript-Dateien (*.sws) können mit jedem handelsüblichen Editor erzeugt werden und beinhalten alle Informationen, die Swift-Generator benötigt, um neue Flash-Dateien zu erzeugen. Die Syntax ist relativ einfach und wird im Folgenden genauestens erklärt. Zudem besteht die Möglichkeit, mit dem Swift-Generator die in eine Vorlage eingefügten Swift-Variablen auszulesen und als Grundlage für das eigentliche Skript zu benutzen, das einige Tipparbeit erspart und somit Tippfehler vermeidet. In diesem Skript stehen alle Anweisungen, die Swift-Generator ausführen soll.

Um Objekte wie z. B. Text, Bilder, Sound etc. zu referenzieren, benutzt Flash intern so genannte tagIDs, die einfach aus einheitlichen (Ident-) Nummern bestehen. Detailliertere Informationen zu der Dateistruktur finden Sie unter www.openswf.org.

Swift-Generator benutzt unter anderem diese tagIDs um Objekte auszutauschen oder zu verändern. Da man diese IDs normalerweise nicht angezeigt bekommt, bietet Swift-Generator die Möglichkeit, diese Informationen aus der Template- oder Flash-Datei auszulesen (Flash File Dump) und als Grundlage für das weitere Swift-Skript zu benutzen.

Zusätzlich können die in der Vorlage eingefügten Swift-Variablen und bei einigen Objekten die Symbol- bzw. Instanznamen angesprochen werden.

Ein Swift-Skript bekommt die Dateiendung sws und beinhaltet als Erstes einen globalen Teil mit allgemeinen Informationen und als Zweites die Anweisungen für Text, Bild, Soundaustausch oder Schriftartdefinitionen etc.

Die Anweisungen werden grundsätzlich groß geschrieben und darauf folgende Parameter oder Variablen klein. Swift-Generator führt diese Anweisungen Schritt für Schritt aus, bei Syntaxfehlern wird die Ausführung gestoppt.

Template-/Flash-Datei auslesen

Um Flash-Dateien auszulesen, startet man Swift-Generator über die Eingabeaufforderung mit der Option –**d.**

```
swiftgenerator.exe -d vorlage.swt > skript.sws
```

Hierbei steht **vorlage.swt** für den Namen der Template- oder Flash-Datei. Es können auch absolute Verweise angegeben werden, z. B. c:\projektx\vorlage.swt.

Am Ende des Befehls wird die Ausgabedatei angegeben (skript.sws). Diese sollte die Endung sws tragen und kann mit jedem Texteditor geöffnet werden.

Das Skript könnte dann folgendermaßen aussehen:

```
% Script template from Template file best_site.swt
INPUT "best_site.swt"
% Output for testing
OUTPUT "export.swf"
% Output for CGI
%OUTPUT -cgi "-"
% Font definitions
% FONT 2 is Arial Bold
SUBSTITUTE TEXT 3 {
FONT 2 HEIGHT 24 COLOR #ffff99
```

```
STRING "Best site of the day"
}
.....
```

Dieses kann man dann nach Bedarf abändern und für die Erzeugung neuer Flash-Dateien benutzen.

Kommentare

Kommentare werden mit dem %-Zeichen eingeleitet und gelten bis zum nächsten Zeilenumbruch, also immer für eine Zeile. Hier das Beispiel:

```
% Dies ist ein Kommentar
```

Globale Angaben

Die globalen Angaben, die in keinem Skript fehlen sollten, sind folgende:

- `INPUT "datei.sw?"`
  `"datei.sw?"` ist der Name der Vorlagendatei (swt oder swf). Die Anführungszeichen verweisen auf die Standardeingabe als Dateneingang.
- `OUTPUT [ -cgi ] [ -expires "datum/+delta" ] "datei.swf"`
  `"datei.swf"` ist der Name der Ausgabedatei. Wird "-" an Stelle eines Dateinamens angegeben, leitet Swift-Generator die Daten an die Standardausgabe.
  `-cgi` stellt der Ausgabe einen HTTP-Header voran (optional, benötigt für CGI-Modus).
  `-expires "datum/+delta"` ist optional, erzeugt eine `"expires"`-Angabe (Gültigkeit der Datei) im HTTP-Header. Es kann ein Datum (im RFC1945-Format) angegeben werden oder die Ablaufzeit `'delta'` in Sekunden mit vorangestelltem Pluszeichen.
- `TTFPATH "fontverzeichnis"`
  Gibt den Pfad zum Verzeichnis mit benötigten Schriftarten (Fonts) an. Diese Angabe ist nur nötig, wenn man Flash-Dateien als Vorlage benutzt oder Schriftarten austauschen möchte.
- `FLASH { FRAMERATE n }`
  In zukünftigen Versionen wird es weitere Parameter für den Flash-Abschnitt geben.
  `n` gibt hierbei die Framerate in Bildern pro Sekunde an (nur ganzzahliger positiver Wert).

- `SAVEDIR "verzeichnispfad"` bzw. `SAVE "datei.swf"`
  Diese Anweisungen werden benötigt, wenn man im CGI-Modus Dateien zusätzlich speichern will. Während bei dem Dateinamen auch Variablen angegeben werden können, muss der Verzeichnispfad als Zeichenkette angegeben werden! Der komplette Speicherpfad wird aus beiden Angaben zusammengesetzt. Enthält der Pfad relative Angaben (*../*), wird der Vorgang aus Sicherheitsgründen abgebrochen.

Schriften und Dateinamen
Wenn man Swift-Generator mit der Option `-f` startet, erhält man eine Liste aller in einem Verzeichnis vorhandenen Schriften und deren Dateinamen. Der Befehl dazu lautet z. B.:

```
swiftgen.exe -f c:\windows\fonts (unter Windows) oder
swiftgenerator -f /pfad/zum/schriftverzeichnis (unter
Linux).
```

Vorsicht
Die Benutzung von SAVEDIR und SAVE kann unter Umständen ein Sicherheitsrisiko bedeuten. Wer sich mit serverseitigen Anwendungen und deren Sicherheitsrisiken nicht auskennt, sollte diese Anweisungen also nicht benutzen.

Variablen benutzen in Swift-Skript

Innerhalb des Swift-Skriptes können Variablen definiert werden. Die Syntax ist relativ einfach: `SET myvar value`

Dies erzeugt eine Variable namens `myvar` mit dem Wert `value`. `Value` kann hierbei eine Zeichenkette, eine Variablenreferenz, das Ergebnis eines Subprozesses (siehe hierzu auch Anweisungen, die Zeichenketten zurückgeben, wie FILE oder DBQ) oder eine Verbindung dieser Werte sein.

Beispiele:

```
SET var1 "Hallo Welt"
```

Der Variablen var1 wird eine Zeichenkette zugewiesen.

```
SET var2 $var1
```

Der Variablen `var2` wird der Wert einer Variablenreferenz (`$var1="Hallo Welt"`) zugewiesen.

Es können auch Bereiche oder einzelne Zeilen einer mehrzeiligen Variablen einer anderen Variablen zugewiesen werden:

```
SET bereich $text<2..3>
```

Würde der Variablen `bereich` die zweite und dritte Zeile der Variablen `text` zuweisen.

```
SET bereich $text<5>
```

Würde der Variablen `bereich` die fünfte Zeile der Variablen `text` zuweisen.

Ein Subprozessergebnis ist ein Kommandozeilenbefehl, der innerhalb von zwei eckigen Klammern angegeben wird:

```
SET datum [ date ]
```

Weist der Variablen `datum` das Ergebnis des Kommandozeilenbefehls `'date'` zu. Ein Kommandozeilenbefehl ist jeder gültige Befehl, der in ein Terminal-/Konsolenfenster oder DOS-Shell eingegeben werden kann.

Schriftarten definieren

Benutzt man Flash-Dateien (swf) als Vorlage, müssen Schriftarten definiert werden, bevor man Text in irgendeiner Art und Weise verändern oder austauschen möchte! Für Template-Dateien (swt) ist dieses nur notwendig, wenn man Schriftarten austauschen möchte.

```
DEFINE FONT id "schriftart.ttf"
```

`id` gibt die `tagID` des Fonts an, der durch »schriftart.ttf« ausgetauscht werden soll. `tagID` und Schriftart müssen existieren, ansonsten wird der Vorgang abgebrochen. Leider kann Swift-Generator nicht mit allen True-Type-Schriften umgehen.

Text modifizieren

Die Syntax für diesen Anweisungsblock lautet:

```
SUBSTITUTE TEXT id {
   ANWEISUNG

}
```

`id` entspricht der `tagID` des zu ändernden Textes der Flash-Vorlage.

Folgende Angaben können innerhalb der geschweiften Klammern gemacht werden:

- `FONT id`
  Ändert die Schriftart des Textes in die Schriftart mit der `tagID` id. Es können nur bereits existierende (also vordefinierte) Font-ids benutzt werden.
- `HEIGHT höhe`
  Setzt die Schriftgröße auf die in `höhe` angegebene Zahl (positiver Integerwert). Die Größe wird in Punkten angegeben.
- `COLOR #RRGGBB`
  Ändert die Schriftfarbe in den angegebenen RGB-Wert (Hexadezimalformat).
- `ALPHA wert`
  Ändert die Transparenz. Es kann ein Wert zwischen 0 und 255 angegeben werden.
- `VSPACING wert`
  Setzt den vertikalen Zeilenabstand eines mehrzeiligen Textes. Muss ein Dezimalwert sein, Vorgabe ist 1.0.
- `KERNING wert`
  Ändert den Abstand zwischen den Buchstaben. Dezimalwert, Vorgabe ist 1.0.
- `STRING [ -nocr ] "text"`
  Definiert einen neuen Text, der mit der aktuellen Schrift, Schriftgröße und -farbe erzeugt wird. Es können mehrere Textdefinitionen hintereinander stehen, die durch einen Zeilenumbruch getrennt werden, außer es wird die Option `-nocr` mit angegeben.
- `ALIGN position`
  Gibt die Ausrichtung des Textes an. Die Ausrichtung richtet sich nach den Grenzen des ursprünglichen Textes. Mögliche Werte sind `left`, `right` und `center`. Vorgabewert ist `left`.

Es können so viele neue Zeichenketten wie nötig angegeben werden, auch wenn der ursprüngliche Text nur einzeilig war.

Wird die Schriftart als Erstes geändert, kann es zu einer falschen Positionierung des Textes kommen. Swift-Generator berechnet die Position zwar neu, kann dieses aber nur zuverlässig tun, wenn die Größe als Letztes angegeben wird.

Textfelder modifizieren

Textfelder können ähnlich wie einfacher Text verändert werden. Die Syntax lautet hier:

```
SUBSTITUTE TEXTFIELD id {
   ANWEISUNG
}
```

`id` verweist auf die `tagID` des Textfeldes, das verändert werden soll.

Folgende Anweisungen können innerhalb des Blockes gemacht werden:

- `FONT id`
  Ändert die Schriftart des Textfeldes zu der Schriftart mit der `tagID` `id`.
- `HEIGHT höhe`
  Ändert die Schrifthöhe des Textfeldes in den Wert `höhe` (Dezimalwert).
- `COLOR #RRGGBB`
  Ändert die Schriftfarbe in den angegebenen RGB-Wert (Hexadezimalformat).
- `ALPHA wert`
  Ändert die Transparenz. Es kann ein Wert zwischen 0 und 255 angegeben werden.
- `STRING "text"`
  Ändert den Inhalt des Textfeldes.
- `ALIGN position`
  Gibt die Ausrichtung des Textes an. Die Ausrichtung richtet sich nach den Grenzen des ursprünglichen Textes. Mögliche Werte sind left, right und center. Vorgabewert ist left.

Die Anweisungen sind nicht kumulativ, die letzte Anweisung überschreibt eine identische vorhergehende.

Sound austauschen

Folgende Sounddateien können ausgetauscht werden:

- WAV-Dateien in 8 oder 16 Bit mit einer Samplerate von 5.5, 11, 22 oder 44 kHz.
- MP3-Dateien mit 11, 22 oder 44 kHz in allen gängigen Bitraten.

Soundobjekte werden über die entsprechende `tagID` angesprochen oder über ihren Symbolnamen.

```
DEFINE SOUND "symbolname" [ -bits n ] "sounddateiname"
```

oder

```
DEFINE SOUND id [ -bits n ] "sounddateiname"
```

`-bits n:` Der optionale Parameter `-bits` legt die Bitkompression fest und kann die Werte 2, 3, 4 oder 5 haben (4 ist Vorgabe). Diese Option wird bei MP3-Dateien ignoriert !

Filmsequenzen austauschen

Diese Anweisung ermöglicht es, einer Filmsequenzinstanz ein anderes Symbol zuzuordnen.

Um eine Filmsequenzinstanz zu erstellen, zieht man das gewünschte Symbol von der Bibliothek auf die Bühne. Dort klickt man die Symbolinstanz mit der rechten Maustaste an und geht dann auf EIGENSCHAFTEN. Nun muss der Symboltyp FILMSEQUENZ ausgewählt und ein Instanzname angegeben werden.

Hier die Anweisung für den Austausch des zur Instanz gehörenden Symbols:

```
DEFINE MOVIE "instanzname" "symbolname"
```

Weist der Instanz `instanzname` das Symbol `symbolname` zu. Existiert das neue Symbol nicht, wird die Instanz nicht verändert.

Weitere Änderungen kann man über folgenden Anweisungsblock erreichen:

```
DEFINE MOVIE "instanzname" {
   ANWEISUNG
...
}
```

Mögliche Anweisungen sind:

- `NAME "symbolname"`
  Name des neuen Symbols.
- `COLOR #RRGGBB`
  Jede Farbkomponente (Rot, Grün und Blau) wird zu den prozentualen (siehe `BALANCE`) Ursprungsfarbwerten addiert. Die Werte werden hexadezimal angegeben, Vorgabe ist #000000 (Schwarz).
- `ALPHA wert`
  Der Alphawert wird zu der prozentualen Ursprungstransparenz addiert. Die Werte gehen von 0 bis 255. Vorgabe ist 0.
- `BALANCE wert`
  Gibt an, wie viel Prozent des Ursprungswertes der beiden vorigen Parameter zu den neuen Werten addiert werden. Wert 0.0 bedeutet 0 % des Ursprungswertes, und Wert 1.0 bedeutet 100 %.

Die Berechnung des resultierenden `COLOR`- bzw. `ALPHA`-Wertes geschieht folgendermaßen:

```
NEUER WERT =  URSPRUNGSWERT x BALANCE + COLOR bzw.ALPHA
```

Bilder austauschen

Swift-Generator kann mit JPEG-, GIF- oder PNG-Dateien umgehen. Bilder werden wie Sounds über die `tagID` oder den Symbolnamen angesprochen, die entsprechende Anweisung lautet:

```
DEFINE IMAGE "symbolname" [ -quality N ] [ -size B,H ]
"bilddateiname"
```

oder

```
DEFINE IMAGE id [ -quality N ] [ -size B,H ] "bilddateiname"
```

`id` bezeichnet die `tagID` des auszutauschenden Bildes.

- `-quality N`
  definiert die Kompressionsrate in Prozent. Für N können Werte zwischen 10 und 100 eingegeben werden. Je höher der Wert, umso höher die Bildqualität und somit umso niedriger die Kompression. Vorgabewert ist 50 %.
- `-size B,H`
  Mit Hilfe dieser Option kann Swift-Generator die Größe eines Bildes anpassen. B steht hier für die Breite und H für die Höhe in Pixeln, auf die das Bild vergrößert oder verkleinert werden soll. Diese Angabe sollte auf jeden Fall gemacht werden, wenn die Abmessungen des neuen Bildes von der Vorlage abweichen, da es sonst zu unschönen Effekten kommen kann.

Daten aus Datenbanken

Seit Version 0.15 unterstützt Swift-Generator auch Datenbanken auf den Plattformen Windows (ODBC), Linux und Mac OS X (MySQL), wodurch dieses Tool einiges an Flexibilität gewonnen hat.

Folgende Syntax wird zum Verbinden mit einer Datenbank verwendet:

```
DATABASE {
   DRIVER "name_des_datenbanktreibers"
   NAME "name_der_datenbank"
   [ HOST "server_adresse" ]
   [ USER "user_id" ]
   [ PASSWORD "password" ]
   [ PORT tcp_port ]
}
```

Die einzelnen Parameter:

- `DRIVER "name_des_datenbanktreibers"`
  Hier muss der Pfad/Dateiname zum MySQL- bzw. ODBC-Datenbanktreiber angegeben werden.
- `NAME "name_der_datenbank"`
  Bezeichnet den Namen der MySQL-Datenbank bzw. bei ODBC-Datenbanken den DSN (Data Source Name).
- `HOST "server_adresse"`
  Optionaler Parameter, der den Servernamen bzw. -adresse definiert. Vorgabewert ist `localhost`. Wird bei ODBC-Datenbanken ignoriert.

- `USER "user_id"`
  Optionaler Parameter, um den Benutzernamen für eine Datenbank anzugeben. Vorgabewert ist die aktuelle User-ID (bei Unix). Entspricht der UIN bei ODBC-Datenbanken.
- `PASSWORD "password"`
  Optionaler Parameter entsprechend zur User-ID. Unter ODBC als PWD bekannt.
  Man sollte darauf achten, dass die Swift-Skriptdatei nicht über HTTP-Requests abrufbar ist, da sonst jeder auf das Passwort zugreifen kann.
- `PORT "tcp_port"`
  Optionaler Parameter für MySQL-Datenbanken. Vorgabewert richtet sich nach dem Datenbanktreiber, bei Problemen sollte man 3306 versuchen. Wird bei ODBC ignoriert.

Kann mit der Datenbank keine Verbindung aufgenommen werden, bricht Swift-Generator den Vorgang ab und gibt eine HTML-Datei mit der Fehlermeldung aus (CGI-Modus). Diese wird allerdings nicht angezeigt, wenn der Aufruf in den EMBED- bzw. OBJECT-Tag einer HTML-Datei eingebettet ist. Testweise sollte man die Funktion also direkt aufrufen.

Datenbankabfrage/Queries

Ist eine Verbindung zur Datenbank zustande gekommen, kann man über die DBQ-Anweisung SQL-Anfragen starten. Hierzu sollte man sich mit der SQL-Syntax auskennen: `DBQ "query"`

Innerhalb der Anführungsstriche stehen die entsprechenden SQL-Anfragen in der SQL-Syntax. Swift-Generator leitet diese weiter an die Datenbank und gibt dann die Ergebnisse zurück. Die Anfragen können auch Verknüpfungen von Zeichenketten und Variablen sein.

Das Ergebnis der Anfrage kann man dann einer Variablen zuweisen, z. B.:

```
SET data DBQ "select ort from adressen where plz=54"
```

Hierzu ein einfaches Beispiel, um z. B. ein Newsforum zu aktualisieren:

```
SET forum "news"
SET id "id"
SET headlines DBQ "select headline from " + $forum + " where id=" + $id
SET title1 $headlines<1>
```

```
SET title2 $headlines<2>
SET title3 $headlines<3>
SUBSTITUTE TEXT 3 {
STRING $title1
}
...
```

Man sollte Skripten mit solchen Anfragen immer vorher über die Kommandozeile testen, da Swift-Generator bei Fehlern die entsprechende Fehlermeldung der Datenbank zurückgibt.

Daten aus Textdateien

Man kann auf einfache Weise Inhalte einer Textdatei auslesen. Die Syntax ist eigentlich selbsterklärend und lautet:

```
FILE "dateiname.txt"
```

Diesen Inhalt kann man dann wie bei der DBQ-Anweisung einer Variablen zuordnen.

```
SET text FILE "dateiname.txt"
```

Referer-Liste

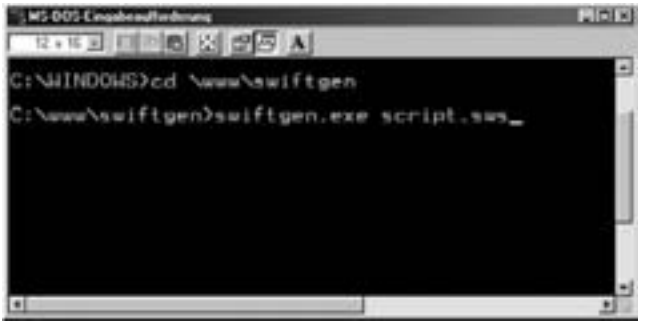
Diese Funktion wird oft in Verbindung mit CGI-Programmen benutzt, wird also nur benötigt, wenn man Swift-Generator im CGI-Modus ausführen will. In der Refererliste werden Domains angegeben, die auf das CGI zugreifen dürfen. Somit schränkt man das Sicherheitsrisiko ein und sollte deshalb auch nicht darauf verzichten. Die Syntax lautet:

```
REFERERS { "www.domainx.com" "localhost" ... }
```

Man kann so viele Domains wie nötig angeben. Die Angabe `"localhost"` ist z. B. nötig, wenn man das Skript auf seinem lokalen Server testen möchte, je nachdem, welcher Alias angegeben wurde.

Wird versucht, von einer anderen Domain das Skript auszuführen, generiert Swift-Generator eine HTML-Nachricht »Access denied« und

Abbildung 4 ▸
DOS-Aufruf



bricht den Vorgang ab. Verwendet man keine Refererliste, kann von jeder Domain aus auf das Skript zugegriffen werden.

Die Refererliste bezieht sich nur auf das jeweilige Skript, in dem es steht.

Kommandozeile oder CGI – online oder offline

Der letzte Schritt erzeugt die eigentliche Flash-Datei mit den neuen Inhalten. Wie am Anfang bereits erwähnt, kann man Swift-Generator offline über Kommandozeilenbefehl ausführen oder online im CGI-Modus. Wie man arbeitet, hängt hauptsächlich davon ab, ob man Zugriff auf den Server hat, wie gut die Performance ist (abhängig vom Inhalt des Filmes) und wie oft der Film aktualisiert werden soll.

Die Offline-Verwendung ist z. B. sinnvoll, wenn man in großer Zahl ähnliche Flash-Dateien erzeugen muss, in denen sich nur einzelne Elemente (z. B. Logos) ändern.

Dateien offline erzeugen

Hier wird einfach über das Terminalfenster das Programm gestartet und als Parameter der Name des Swift-Skriptes mit angegeben. Unter Windows z. B. so, wie in Abb. 3 gezeigt.

Man kann Swift-Generator natürlich auch von anderen Programmen aus starten, was z. B. in Verbindung mit der Erzeugung von Standard-HTML-Seiten Sinn haben kann.

Dateien online erzeugen

Online kann man Swift-Generator per HTTP-Aufruf starten, auch hierbei muss man den Namen des Skriptes als Parameter mit übergeben. Das Ganze sieht dann z. B. so aus:

http://www.domainxyz.de/cgi-bin/swiftgen.exe?sws=skript.sws

Damit Swift-Generator die Flash-Datei an den Browser zurückgibt, muss im globalen Teil des Swift-Skriptes die Anweisung `OUTPUT -cgi "-"` stehen.

Sinnvollerweise kann man den Aufruf auch wie eine richtige Flash-Datei direkt in die OBJEKT-/EMBED-Tags einer HTML-Datei einsetzen.

Swift-Generator auf dem Server installieren

Für Swift-Generator benötigt man einen Provider, der es erlaubt, selbstausführbare CGI-Programme laufen zu lassen. Leider erlauben viele ISPs nur vorinstallierte CGIs oder nur CGI-Skripten wie Perl oder PHP. Hier muss man sich also vorher bei seinem Provider erkundigen.

Wer Swift-Generator auf seinem lokalen Rechner testen möchte, muss dazu einen Webserver installieren. Ich selber habe mit Omni-HTTPd (www.omnicron.ab.ca) ganz gute Erfahrungen gemacht, Apache (www.apache.org) ist aber auch sehr beliebt (allerdings etwas schwieriger zu administrieren).

Hat man jetzt einen passenden ISP oder seinen lokalen Webserver eingerichtet, muss man die Binärdatei für das richtige Betriebssystem (Linux, Windows oder Mac ...) von http://www.swift-tools.com/swift-generator.html herunterladen.

Die entpackten Dateien sind je nach Betriebssystem unterschiedlich. Unter Linux findet man eine Datei namens swift-generator, die man in swift-generator.cgi umbenennen sollte. Unter Windows heißt die Datei swiftgen.exe, deren Endung nicht umbenannt werden darf. Theoretisch kann man diese Dateien jetzt einfach in das Verzeichnis /cgi-bin des Webservers kopieren, unter Linux noch die Dateirechte auf z. B. chmod 755 setzen (-rwxr-xr-x), und das Programm sollte laufen.

Zum Testen kann man das Programm mit http://www.domainname.de/cgi-bin/swiftgen.exe oder lokal mit http://localhost/cgi-bin/swiftgen.exe aufrufen. Als Antwort sendet Swift-Generator Folgendes:

Swift Generator
Version ?.? Copyright 1999–2000
Olivier Debon

Abbildung 5 ▸
Standard-CGI

Swift-Generator im eigenen Verzeichnis

Die Swift-Skriptdateien müssen im selben (bzw. untergeordneten) Verzeichnis liegen wie das Programm selber. Das Verzeichnis /cgi-bin ist aber eigentlich nicht der richtige Ort für solche Dateien. Wenn man die Möglichkeit dazu hat bzw. der ISP einwilligt, sollte man ein eigenes Verzeichnis für Swift-Generator anlegen. Ich benenne es immer mit /swiftgen.

Damit Swift-Generator in diesem Verzeichnis läuft, muss man dem Webserver dieses noch mitteilen. Unter Apache trägt man dazu in der Datei access.conf (bzw. httpd.conf) Folgendes ein:

```
<Directory /pfad/zum/verzeichnis/swiftgen >
AllowOverride None
Options ExecCGI
</Directory>
```

Damit der Server Dateien mit der Endung .cgi auch als CGI-Dateien interpretiert, muss in der Datei srm.conf noch Folgendes ergänzt werden:

```
AddHandler cgi-script .cgi
```

Benutzt man Omni-HTTPd geht man unter Properties·Default Virtual Settings in die Rubrik Standard-CGI. Dort kann man das neue Verzeichnis mit einem entsprechenden Alias angeben.

Bekannte Probleme

Hier noch ein paar Informationen zu bekannten Problemen und deren Lösung.

Mac OS

Einige auf Mac OS erstellte Template- oder Flash-Dateien enthalten Schriftartinformationen, die Swift-Generator nicht richtig interpretieren kann. Man sollte in diesem Fall die Vorlagen unter Windows erstellen.

Internet Explorer

Microsoft Internet Explorer (IE) hat einen Bug, wenn ein über CGI generiertes Multimediaobjekt in einem Vollbild (100 % im Fenster oder Frame) angezeigt werden soll. Wird Swift-Generator direkt (z. B. über die Adresszeile) aufgerufen, kann dadurch der Explorer abstürzen. Serverseitige Analysen ergaben, dass er mehrere Anfragen hintereinander abschickt (mit fehlerhaften Parametern in der letzten Abfrage). Dadurch wartet der Explorer immer noch auf zurückkommende Daten, obwohl bereits alle gesendet wurden.

Ist der Aufruf hingegen in ein OBJECT-Tag eingebunden, funktioniert es ohne Probleme. Auch unter Netscape läuft beides problemlos.

Hier ist ein kleines Perl-Skript, welches das Problem umgeht, der Aufruf erfolgt dann mit `embed.cgi` an Stelle von `swift-generator.cgi`. Das Skript packt den Aufruf in HTML-Code und leitet ihn dann einfach weiter an Swift-Generator. Eventuell muss das Skript noch ein bisschen angepasst werden.

```
embed.cgi:
#!/usr/bin/perl
print "Content-Type: text/html\r\n\r\n";
if ("$ENV{REQUEST_METHOD}" eq 'POST') {
$query = <>;
$query =~ s/(\r|\n)//g;
} else {
$query = $ENV{QUERY_STRING};
}
print <<END_OF_HTML;
```

```
<HTML>
<BODY BGCOLOR=#ffffff>
<OBJECT CLASSID="clsid:D27CDB6E-AE6D-11cf-96B8-444553540000"
WIDTH=100% height=100%>
<PARAM NAME="MOVIE" VALUE="swift-generator.cgi?$query">
<PARAM NAME="QUALITY" VALUE="HIGH">
<EMBED SRC="swift-generator.cgi?$query" WIDTH=100% HEIGHT=100%
TYPE="application/x-shockwave-flash">
</EMBED></OBJECT>
</BODY>
</HTML>
END_OF_HTML
```

Probleme mit Webservern

Mit Apache (Unix und Windows), Sambar Server (Windows) und OmniHTTPd (Windows) wurden bisher gute Ergebnisse erzielt. Einige Webserver halten sich allerdings nicht an CGI1.1-Konventionen.

Bei manchen Servern (Xitami, IIS, Personal Webserver) wird die Testseite nicht angezeigt, wenn Swift-Generator ohne Parameter aufgerufen wird. In diesem Fall kann man einfach den Parameter `?test` an den Aufruf anhängen (`swiftgen.exe?test` oder `swift-generator.cgi?test`).

Viele Probleme liegen aber einfach an einer falschen Serverkonfiguration oder falsch vergebenen Dateirechten (Unix). Also bitte immer erst die Serverdokumentation gründlich durchlesen.

Arbeitet man mit Microsofts Internet Information Manager (IIS), sollte man auf keinen Fall `swiftgen.exe` in `swiftgen.cgi` umbenennen, da es sonst Probleme mit eventuell installierten Perl-Skripten geben könnte.

IIS startet CGI-Dateien immer aus dem Hauptwebverzeichnis. Arbeitet man mit den Voreinstellungen, ist dieses C:\InetPub\wwwroot. Wurde die Swift-Skriptdatei skript.sws in das Unterverzeichnis C:\InetPub\wwwroot\swiftgen gelegt, wäre der Aufruf, um Swift-Generator zu starten, entsprechend:

```
http://hostname/cgi-bin/swiftgen.exe? sws = swiftgen/skript.sws
```

Anwendungsbeispiele für serverseitige Skripts

Flash, PHP & MySQL

▸ Peter Karsten, www.peter-karsten.de

Wer mit Flash und Variablen aus Textdateien gearbeitet hat, weiss, dass man mit den Textdateien schnell an ein Limit gerät und die Informationen nicht so darstellen kann, wie man sie gern hätte. Was kann man also mit der Skriptsprache PHP machen? Viele Beispiele gibt es noch nicht, die dies zeigen.

Um eine Website ständig aktuell zu halten, benötigt man aktuelle Informationen. Diese Informationen zu pflegen, bedarf einiges an Aufwand, daher wird es immer üblicher, Internetseiten mit Informationen zu versorgen, die aus einer Datenbank stammen. Eine Datenbank zu pflegen ist nicht sehr schwer, und wenn man ein bisschen geschickt ist, kann man sogar als unerfahrener User die Datenbestände eines Webangebotes aktualisieren. Da mir die Skriptsprache PHP ans Herz gewachsen ist, ich aber auch Flasher bin, möchte ich anhand eines Beispiel zeigen, wie eine solche Zusammenarbeit möglich ist. Um dies zu bewerkstelligen, stütze ich mich auf ein Beispielskript, das ich für Powerflasher entwickelt habe. Es kann eine MySQL-Datenbank nach Postleitzahlen durchsuchen und die dazugehörigen Ergebnisse in Flash ausgeben.

Erstellung der MySQL-Datenbank

Für Benutzer, die noch keine Erfahrung im Umgang mit PHP und MySQL haben, empfehle ich meine Lieblingsseiten: www.php-center.de und www.little-idiot.de/mysql, wo man einen recht schnellen Überblick und Einstieg in diesen Bereich bekommt.

Die Struktur der hier im Beispiel benutzten Datenbank sieht wie folgt aus:

```
CREATE TABLE plz (
id int(11) DEFAULT '0' NOT NULL auto_increment,
PLZ varchar(11) NOT NULL,
adresse varchar(100) NOT NULL,
plzort varchar(30) NOT NULL,
telfon varchar(40) NOT NULL,
telfax varchar(40) NOT NULL,
email varchar(80) NOT NULL,
PRIMARY KEY (id)
);
```

Diese Struktur habe ich aus einem MySQL-Dump entnommen, genauer kann man die Struktur aber auch in tabellarischer Form darstellen.

Struktur der Datenbank

Feldname	Feldtyp	Feldgröße	Extra
id	int	11	auto_increment
plz	varchar	6	
adresse	varchar	100	
plzort	varchar	30	
telfon	varchar	40	
telfax	varchar	40	
email	varchar	80	

Damit wäre der erste Schritt geschafft. Den zweiten Schritt, wie man PHP-Dateien erstellt, überspringen wir, da wir ein Flash-Buch schreiben. Dennoch werde ich auch für den folgenden Punkt das Skript abbilden, aber nur das Wesentliche dazu erklären. Mehr dazu auf den oben genannten URLs.

Ausgabe der Datenbankinhalte in eine PHP-Datei

Um die PHP-Datei zu erstellen, wird nur ein einfacher Texteditor benötigt. Je nachdem, wie der Server konfiguriert ist, hat diese Datei die Endung *.php, *.php3,*.php4 oder auch andere mögliche Varianten. Die Datei (ich nenne sie plz.php3) sieht dann so aus:

```
<?php
```

Hier wird die Datenbankverbindung zu der MySQL-Datenbank aufgebaut. `localhost` steht für den Servernamen, `dbname` steht für den Namen, den die Datenbank trägt, User und Password sind selbsterklärend.

```
$link = mysql_connect("localhost","user","password");
mysql_select_db("dbname",$link);
```

Hier wird geprüft, ob die Eingabe des Formularsuchfeldes einen Wert enthält oder nicht. Wenn das Suchfeld ohne Wert übermittelt wurde, so erscheint dann die Meldung, dass keine PLZ gefunden wurde. Wenn Text oder sonstige Zeichen eingegeben werden, so erscheint die Meldung, dass keine Treffer gefunden wurden. Man könnte dies verfeinern, indem man prüft, ob nur Zahlen eingegeben wurden und in welcher Anzahl, aber das ist jedem selbst überlassen ...

```
if ($plz=="") {
   echo "quote=Sie haben keine PLZ ausgewaehlt";
}
else
```

Dieses Skript ist so gedacht, dass man die Anfangszahlen einer PLZ eingeben kann, sprich es wird nach einem Gebiet gesucht. Wenn man z. B. 41 eingibt, so werden alle Einträge gelistet, deren PLZ mit 41 beginnt; wird 41564 eingegeben, so erscheinen nur die Einträge mit passender PLZ.

Es müssen immer zwei Ziffern angegeben werden, weil die Spalte »plz« in der Datenbank nur zwei Ziffern enthält, die das jeweilige PLZ-Gebiet darstellen sollen. Der Ausdruck zählt die ersten beiden Ziffern des eingegebenen Wertes und übernimmt diese als Variable.

```
$plz = substr ("$plz",0,2);
```

Dies ist die eigentliche Datenbankabfrage.

```
$sql = "select PLZ,regionalleitung,adresse,telfon,
telfax,email from orte where plz = '$plz' order by plz";
```

Dies ist das Resultat der Datenbankabfrage

```
$result=mysql_query($sql,$link);
$num = mysql_num_rows($result);
```

Wenn die Abfrage Treffer erzielte:

```
if ($num != "0") {
```

Hier kommt der erste Punkt, der in Flash nachher relevant sein wird, die Variablenausgabe. Ich bin hier so vorgegangen, dass ich die ganzen Einträge der Datenbank in eine Variable übergebe. Natürlich könnte man auch jeden einzelnen Wert in einer eigenen Variablen übergeben.

```
echo "&variable=";
```

Hier beginnt die Schleife, welche die Datensätze ausliest und als Variable speichert.

```
while ($rows = mysql_fetch_array($result)) {
echo "PLZ Gebiet: $rows[PLZ]\n".
"$rows[adresse]\n".
"$rows[plzort]\n".
"Telefon: $rows[telfon]\n".
"Telefax: $rows[telfax]\n".
"E-Mail: $rows[email]\n".
"-----------------------------------";
} // Ende Schleife
} // Ende if num Abfrage
```

Falls keine Datensätze gefunden wurden:

```
if ($num == "0") {
echo "&variable=Keine Treffer!!";
}
}
?>
```

Abbildung 1 ▶
Keyframes

Jetzt haben wir die Datenbank eingerichtet und das PHP-Skript geschrieben. Zum Testen reicht das vollkommen aus. Ich persönlich empfehle, das Skript erst auf dieser Basis zu testen, so kann man eventuell auftretende Fehler schon im Skript ausschließen. Die Ausgabe dieses Skripts würde eine ähnliche Liste wie folgt erzeugen:

Für plz.php3 folgende Ausgabe:

```
variable=Sie haben keine PLZ ausgewaehlt
```

Weil kein Wert bzw. keine PLZ übergeben wurde, sollte folgender Anhang an der Datei ein wenig Klarheit schaffen:

`plz.php3?plz=41564` ergibt folgende Ausgabe:

```
variable=PLZ Gebiet: 41 Leitung: Flashstadt, MMStr.. 92-94 MMStr.
415674 Traumland Telefon: 0123 / 123456 Telefax: 0123/ 123457 E-Mail:
```

Alle Daten sind nun in der Variablen (hier `variable` genannt) enthalten und müssen an Flash übergeben werden.

In diesem Beispiel wurden zehn Keyframes verwendet.

Das erste Keyframe enthält das Formularfeld: ein Textfeld, das »plz« genannt wird, dazu ein Button, der mit der Action

```
On (Release)
   Go to and Play (5)
End On
```

versehen ist; die Eigenschaften des ersten Keyframes sind wie folgt notiert:

```
Set Variable: "plz" = ""
Stop
```

Die Variable **»plz«** wird hier geleert, falls man eine neue Suche starten möchte. Andernfalls würde, so wie Flash aufgebaut ist, die erste eingegebene PLZ als Value wiedergegeben.

Beispiel: Wir geben die PLZ 41564 ein und drücken SUCHEN. Dann sind wir im fünften Keyframe angelangt, in diesem Keyframe ist ein einfacher Text wie »Datenbank wird gelesen« hinterlegt. In diesem Keyframe befinden sich die Aktionen:

```
Set Variable: "test" = "plz.php3? plz="
Set Variable: "plz" = plz
Set Variable: "p" = test&plz
Load Variables (p, 0, vars=POST)
```

Die letzte Zeile liest die ausgegebenen Variablen in Flash ein; da wir aber nicht wissen, welchen Wert diese Variable hat bzw. dies nicht vor der Eingabe der PLZ wissen, müssen wir mit Flash der Variablen einen Wert zuweisen. Dieser Wert ist dann unser Suchkriterium für die eventuellen Ergebnisse. Wir müssen dem PHP-Skript mitteilen, dass nach einem Wert gesucht wird, der dann wieder ausgelesen wird. Um das zu verstehen, greife ich noch einmal auf das eben genannte Beispiel zurück:

```
plz.php3?plz=41564
```

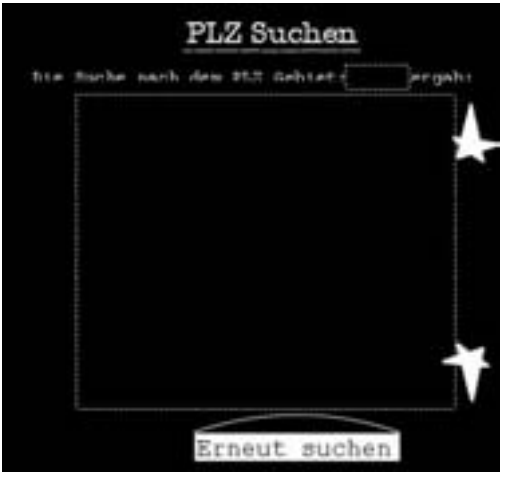
Die erste Zeile (`SET Variable: "test" = "plz.php3? plz=" =`) gibt eine Variable zurück, die folgende Zeichen enthält: `plz.php3?plz=`

Die zweite Zeile setzt die Variable `plz` mit dem im ersten Keyframe zugewiesenen Wert, in unserem Beispiel die: `41564`. Die dritte Zeile macht nichts anderes, als eine Variable zu setzen, welche die ersten beiden zusammenfügt, also: `plz.php3?plz=41564`. Und die vierte Zeile liest die Daten in Flash ein, die zu der dritten Variable passen.

Das sechste Keyframe dient dazu, zu prüfen, ob die Variable bzw. der Text der Variablen vollständig geladen wurde; wenn die Daten geladen wurden, dann soll es weiter zum Frame 10 gehen, ansonsten soll er so lange Keyframe 6 abspielen, bis die Daten vorhanden sind.

```
If (eval(variable) <> "")
Go to and Play (10)
Else Go to and Play (6)
End If
```

Abbildung 2 ▶
PLZ-Suche



Im zehnten Keyframe wird ein Stopper gesetzt, damit der Film nicht überläuft. In diesem Frame muss ein Textfeld erstellt werden, das den Namen »variable« trägt, oder wie auch immer wir die Variable genannt haben, in der die Ergebnisse gespeichert werden. Wir erstellen ein weiteres Textfeld mit dem Namen »plz«, um z. B. dem User mitzuteilen, nach was für einer PLZ er gerade sucht bzw. nach welcher PLZ er gesucht hat.

Da wir nicht wissen, wie lang der Text ist, der in der Variablen gespeichert ist, kann das Textfeld, das wir erstellt haben, schnell an seine Grenzen stoßen. Zu viel Text wird dann einfach nicht angezeigt, deswegen sollten wir uns eines Scrollbalkens bedienen.

Um den Scrollbalken zu erstellen, müssen wir einen MovieClip anlegen, der zwei Buttons enthält (einer scrollt nach oben und der andere hinunter). Der erste Button erhält folgende Action als Eigenschaft:

```
On (Press, Drag Over, Drag Out)
   Set Variable: "/:variable.scroll" = /:variable.scroll-1
End On
```

Der zweite Button erhält als Action folgende Eigenschaft:

```
On (Press, Drag Over, Drag Out)
   Set Variable: "/:variable.scroll" = /:variable.scroll+1
End On
```

Das Ganze sieht dann in etwa so aus wie in Abbildung 1 gezeigt.

Damit wäre auch schon das Wesentliche gesagt, und das Beispiel sollte funktionieren.

Flash und ASP

▸
Michael Schinkel
www.vision4net.de

Die Verbindung Flash und ASP (Active Server Pages) schafft die Möglichkeit, auf einfache und auch kostengünstige Art und Weise dynamisch generierte Textinhalte in Flash-Seiten einzubinden, ohne diese wie sonst mit viel Arbeitsaufwand aktualisieren zu müssen.

In Flash haben Sie eigentlich nicht viele Möglichkeiten, Textinhalte aktuell zu halten. Zum Beispiel ist dies ja mit Textdateien möglich, aus denen die gewünschten Variablen geladen werden. Eine weitere Möglichkeit ist es, einen Generator einzusetzen (z. B. Macromedia Generator, Swift Generator). Dies ist aber meist gar nicht notwendig, zumindest solange es sich nur um Zeichen und nicht um Grafiken handelt. Wenden wir uns nun einer einfacheren Methode zu.

Als Voraussetzung für den Einsatz von Flash und ASP sollten Sie einen Server mit dem Betriebssystem Windows NT sowie den IIS als Webserver zur Verfügung haben.

Die möglichen Anwendungen sind zahlreich, z. B. können Sie Daten, die der User auf Ihrer Flash-Seite eingegeben hat, in einer Datenbank speichern, bei Online-Spielen die Spielergebnisse oder einen Login für Ihre Flash-Seiten realisieren. Wir möchten hier zunächst auf den Bezug von Texten aus der Datenbank in einen Flash-Film eingehen.

Zum Erstellen einer ASP-Seite verwenden wir der Einfachheit halber Macromedia Dreamweaver Ultradev, zum Erstellen der Datenbank wird Microsoft Access verwendet.

Wir beginnen mit der Datenbank; hierzu öffnen wir eine leere Datenbank und erstellen eine neue Tabelle mit fünf Spalten. Die Spalten benennen wir alphabetisch von A bis E. Als Nächstes muss ein System-DSN für diese Datenbank erstellt werden, wir benennen ihn mit »usr_flash.5«.

Eine vorbereitete Datenbank finden Sie auf der Buch-CD.

Mit Macromedia Dreamweaver Ultradev erstellen wir nun eine Datenbankabfrage, nachdem wir eine QDBC-Verbindung zur Datenbank erstellt haben. Auch diese Abfrageseite finden Sie auf der Buch-CD.

Diese Abfrage filtert nach dem URL-Parameter ID den richtigen Datensatz aus.

Abbildung 3 ▶
System-DSN

Damit Flash mit dieser Abfrage etwas anfangen kann, modifizieren wir die .asp-Datei in einem HTML-Editor. Zunächst entfernen wir die HTML-Tags, anschließend wird der Gesamtausdruck der Abfrage nach folgendem Schema korrigiert:

```
ID=...&A=...&B=...&C=...&D=...&E=...
```

Die Punkte stehen hier als Ersatz für den Quelltext der Datenbankabfrage, die von Dreamweaver Ultradev hier bereits eingefügt wurde.

Alle mit der Aktion `loadVariables` geladenen Variablen müssen im Standard-MIME-Format »application/x-www-urlformencoded« sein.

Die Datei kann eine beliebige Anzahl an Variablen enthalten. Variablen- und Wertepaare müssen durch ein kaufmännisches Undzeichen (&), Wörter innerhalb eines Wertes durch ein Plus (+) voneinander getrennt sein.

Nachdem nun alles vorbereitet ist, wenden wir uns endlich dem Flashmovie zu.

Wir erstellen nun eine einfache Seite mit Textfeldern, welche die Namen unserer zuvor verwendeten Variablen tragen. Für diese Textfelder wählen wir die Option Dynamischer Text. Dem ersten Schlüsselbild weisen wir die Aktion `loadVariables` zu. Als Datei geben wir die .asp-Datei (abfrage.asp) an und hängen den URL-Parameter `?ID=1` an (`loadVariablesNum ("abfrage.asp?ID=1", 0);`).

So, nun ist es bis zum ersten Test nicht mehr weit. Der Film kann nur über den Internet-Server auf seine Funktion getestet werden. Mit der

◄ **Abbildung 4**
Modifizieren der .asp-Datei

◄ **Abbildung 4**
Die Flash-Ausgabe

Funktion »Film testen« erreichen Sie nur Fehlermeldungen, da die .asp-Datei nicht ausgeführt werden kann.

Spielen Sie die Dateien auf Ihre Serverumgebung und rufen die index.html mit dem Flash-Film auf. Nun sollte Ihnen in den Textfeldern der mit der ID=1 verbundene Datensatz angezeigt werden. Wenn Sie die ID im `loadVariables`-Befehl verändern, wird der jeweilige Datensatz (2, 3, 4 usw.) angezeigt, so können Sie z. B. über eine Abfrage oder verschiedene Menübuttons die jeweils unterschiedlichen Datensätze aufrufen. Zur Administration Ihrer nunmehr dynamischen Flash-Seite bauen Sie sich mit Dreamweaver Ultradev einige Seiten, mit denen Sie die Datensätze ändern oder erweitern können.

E-Mail-Versand mit ASP

▶ Mario Mies

Das folgende Tutorial erläutert, wie Sie mit Flash 4 und 5 ein Formular per ASP-Skript versenden können; dabei richtet sich das Tutorial an jene Anwender, die einen NT-Provider mit JMail-Unterstützung verwenden. Dieses Skript bietet größte Flexibilität, die einzelnen Einstellungen werden weiter unten beschrieben.

Nun zur Flash-Programmierung: Die einzelnen Daten werden in dem Flash-Formular per `GetURL` und der Aktion `POST` mit der aufgerufenen URL verknüpft und an die Datei sendmail.asp übergeben, ähnlich wie bei einem HTML-Formular. Sie können so viele Felder anlegen, wie Sie benötigen, sie müssen diese dann allerdings in dem ASP-Skript auch festlegen. Es werden einfache Eingabeprüfungen unterstützt, diese können bei Bedarf weiter ausgebaut werden. Die .fla-Datei besteht aus drei Keyframes auf der Hauptbühne. Diese werden wir folgt formatiert:

1. Keyframe

Erstellen Sie beliebige Texteingabefelder, benennen Sie die einzelnen Textfelder (unter den Eigenschaften). Sie benötigen zudem ein Textfeld für den Warnhinweis bei Nichteingabe, nennen Sie dieses z. B. `"txt_out"`. Erstellen Sie im ersten Frame einen Senden-Knopf mit folgendem ActionScript:

```
On (Release)
   If (Name eq "" or Text eq "" or Betreff eq "" or usw. für jedes
Feld, das Sie anlegen)
      Set Variable: "txt_out" = "Bitte geben Sie die erforderlichen
Daten ein"
   Else
      Call ("valid")
      Go to Next Frame
   End If
End on
```

Hier werden die einzelnen Felder auf Eingabe überprüft, bei Nichteingabe erscheint in einem Textfeld ein Warnhinweis. Mit `call ("valid")` wird das Script zum Überprüfen der E-Mail-Adresse aktiviert. Name, Text und Betreff sind in diesem Fall die Namen der Textfelder. Um weitere Felder zu prüfen, müssen die Namen nur in der If-Zeile ergänzt werden.

Erstellen Sie jetzt einen ZURÜCKSETZEN-Knopf zum eventuellen Löschen des Formulars und geben folgendes ActionScript unter den Eigenschaften ein:

```
On (Release)
   Go to and Stop (1)
   Set Variable: "name" = ""
   Set Variable: "email" = ""
   Set Variable: "text" = ""
End on
```

Für jedes angelegte Feld in dem Flash-Film muss eine leere Variable gesetzt werden, um das Formular zu löschen.

Jetzt müssen Sie noch eine `Stop`-Aktion im ersten Keyframe einfügen.

2. Keyframe

Als Erstes müssen Sie den SENDEN-Knopf löschen. Dies ist erforderlich, damit die E-Mail-Überprüfung bei Fehlern mehrfach ausgeführt werden kann, indem man zurück in den ersten Frame springt und das E-Mail-Skript erneut aufruft.

Ändern Sie den Text des ZURÜCKSETZEN-Knopfes in ZURÜCK und geben folgendes Skript unter den Eigenschaften des Knopfes ein:

```
On (Release)
   Go to Previous Frame
   Set Variable: "txt__out" = ""
End On
```

Hier springen Sie in den ersten Frame und löschen den Inhalt des ersten Warnhinweisfeldes.

Erstellen Sie nun ein Textfeld für den zweiten Warnhinweis, und nennen Sie dieses z. B. »txt_out«.

Unter den Eigenschaften des zweiten Keyframes geben Sie Folgendes ein:

```
Set Variable: "i" = "0"
Set Variable: "txt__out" = ""
Loop While (i < Length (Email))
   If (Substring ( Email, i, 1) eq "@" )
```

```
      Set Variable: "positionat" = I
   End If
   If (Substring ( email, i, 1) eq ".")
      Set Variable: "positionpunkt" = i
   End If
   Set Variable: "i" = i + 1
   End Loop
   If (positionpunkt<=positionat or positionpunkt = 0 or positionat = 0
   or (positionpunkt - positionat) < 2 or (Length (Email) - posi-
tionpunkt) < 2 or (Length (Email) - positionpunkt) > 3 or positionat
= 1)
   Set Variable: "txt_out" = "Ungültige E-Mail-Adresse"
   Else
   Go to and Stop (3)
End If
```

Dies ist das Skript zur Überprüfung der E-Mail-Adresse, das Vorhandensein eines @-Zeichens sowie eines ».« wird überprüft. Die Schreibweise »Text@Text.Domain« wird ebenfalls geprüft, ist eine dieser Bedingungen falsch, wird ein Warnhinweis ausgegeben. Ansonsten wird der dritte Frame aufgerufen.

3. Keyframe

Unter den Eigenschaften des Keyframes geben Sie Folgendes ein:

```
Stop
Get URL ("sendmail.asp", window="_blank", vars=POST)
Set Variable: "name" = ""
Set Variable: "email" = ""
Set Variable: "text" = ""
//usw. für jedes Textfeld, das Sie angelegt haben.//
Go to and Stop (1)
```

Mit `GetURL` wird die Skript-Datei aufgerufen. Durch die Aktion `vars=Post` werden die Formulardaten mit der URL verknüpft und an ein neues Browserfenster SENDMAIL.ASP übergeben. Weiterhin wird das Formular gelöscht und der erste Keyframe wieder aufgerufen. Das Formular ist nun wieder bereit für eine neue Eingabe.

Zum ASP-Skript

Das gesamte Skript wird mit HTML-Elementen integriert in einer Datei (sendmail.asp) zusammengefasst. Sie dient dazu, die vom Flash-Film übergebenen Daten zu übernehmen und über den Mail SMTP-Server an den jeweiligen Empfänger zu versenden sowie eventuelle Bestätigungen anzuzeigen; dabei kann auch ein in die Datei integrierter Flash-Film als Bestätigung dienen.

Die einzelnen Formularwerte werden in Variablen gespeichert, die am Anfang des Skripts festgelegt werden. Für jedes in Flash angelegte Formularfeld muss in dem Skript eine Variable definiert werden. Die Formularvariablen für den E-Mail-Header werden weiter unten zum Versand formatiert, dabei ist die Syntax abhängig von der JMail Software. Bei der Formatierung des E-Mail-Bodys werden die vorher festgelegten Variablen aufgerufen und zum Versand formatiert. Hier muss auch für jedes Formularfeld ein Eintrag in der vorliegenden Syntax erfolgen.

Der weitere Inhalt hinter den %> Zeichen ist frei konfigurierbar. Die Datei enthält ein JavaScript, das die Fenstergröße verändert, die Größe kann durch Ändern der Einstellungen (Breite und Höhe) noch verändert werden. Ein Pop-up-Fenster im gewöhnlichen Sinne ist mit Flash nicht möglich, da sonst die Variablen nicht mit übergeben werden können.

Folgende Einstellungen in der Datei sendmail.asp sind vorher zu machen:

```
JMail.ServerAddress = "Mailserver Adresse
```

Hier geben Sie die Mail-Server-Adresse ein, fragen Sie dazu Ihren Provider.

```
Empfaenger = "E-Mail des Empfängers"
```

Geben Sie hier die Empfänger-E-Mail-Adresse ein.

```
Betreff= "Text " & Request.Form ("Betreff")
```

Geben Sie zwischen den "" einen statischen Betreff ein, dieser erscheint immer bei der Mail. Der restliche Betreff wird von dem Formularfeld BETREFF übernommen und erscheint ebenfalls in der Betreffzeile der Mail. Bitte achten Sie auf die Leerzeile vor dem zweiten ". Sie können den Text auch weglassen, indem Sie `"Text" &` löschen, dann wird immer der Betreff des Formulars angezeigt.

```
JMail.Priority = 1
```

Hier können Sie die Wichtigkeit der Mail festlegen: 1 bis 5, von Sehr wichtig bis Normal.

Festlegen der Formularvariablen

```
Name= Request.Form("Name")
```

Variable zur Speicherung der Werte des Flash-Formularfeldes »Name«, dieser erscheint dann als Absendername in der E-Mail.

```
Email= Request.Form("Email")
```

Variable zur Speicherung der Werte des Flash-Formularfeldes »Email«, diese wird mit dem Namen verknüpft und erscheint im Header.

```
Strasse=Request.Form("Strasse")
```

Variable zur Speicherung der Werte des Flash-Formularfeldes »Strasse«, der Wert erscheint im Body der Mail.

```
PLZ=Request.Form("PLZ")
```

Variable zur Speicherung der Werte des Flash-Formularfeldes »PLZ«, der Wert erscheint im Body der Mail.

```
Ort=Request.Form("Wohnort")
```

Variable zur Speicherung der Werte des Flash-Formularfeldes »Wohnort«, der Wert erscheint im Body der Mail.

```
Land=Request.Form("Land")
```

Variable zur Speicherung der Werte des Flash-Formularfeldes »Land«, der Wert erscheint im Body der Mail.

```
Vorwahl=Request.Form("Vorwahl")
```

Variable zur Speicherung der Werte des Flash-Formularfeldes »Vorwahl«, der Wert erscheint im Body der Mail.

```
Rufnummer=Request.Form("Rufnummer")
```

Variable zur Speicherung der Werte des Flash-Formularfeldes »Rufnummer«, der Wert erscheint im Body der Mail.

```
Text=Request.Form("Text")
```

Variable zur Speicherung der Werte des Flash-Formularfeldes »Text«, der Wert erscheint im Body der Mail.

```
Empfaenger= "E-Mail-Adresse des Empfängers"
Betreff= "Text " & Request.Form ("Betreff")
```

Wie oben beschrieben:

```
Set JMail =Server.CreateObject ("JMail.SMTPMail")
```

Hier wird der SMTP Mail-Server initialisiert.

```
JMail.ServerAddress= "Mailserver Adresse"
```

Wie oben beschrieben, wird hier der ausgehende Mail-Server definiert.

Formatierung zur Versendung

```
JMail.SenderName = Name
```

Hier wird der Absendername zur Versendung formatiert, die Variable hinter dem = entspricht der Variablen im oberen Übernahmeteil, Achtung, keine "" eingeben.

```
JMail.Sender= Sender-E-Mail
```

Hier wird die E-Mail des Absenders zur Versendung formatiert, die Variable hinter dem = entspricht der Variablen im oberen Übernahmeteil; Achtung, keine "" eingeben.

```
JMail.Subject= Betreff
```

Hier wird der Betreff zur Versendung formatiert, die Variable hinter dem = entspricht der Variablen im oberen Übernahmeteil; Achtung, keine "" eingeben

```
JMail.AddRecipient Empfaenger
```

Hier wird die Empfängeradresse zur Versendung formatiert, die Variable entspricht der Variablen im oberen Übernahmeteil; Achtung, keine "" eingeben.

Formatierung des E-Mail-Bodys

```
JMail.Body="Text"&vbCrLf&"Name: "&Name&vbCrLf&"E-Mail:
"&Email&vbCrLf&"Strasse: "&Strasse&vbCrLf&"PLZ: "&PLZ&vbCrLf&"Ort:
"&Ort&vbCrLf&"Land: "&Land&vbCrLf&"Vorwahl: "&Vorwahl&vbCrLf&"Ruf-
nummer: "&Rufnummer&vbCrLf&"Text: "&Text
```

Unter "Text" kann ein statischer E-Mail-Body-Text stehen, dieser wird dann immer bei der Mail angezeigt; mit vbcrlf erfolgt ein Zeilenumbruch, weiterhin werden die einzelnen »Überschriften« der Formularwerte, die in der E-Mail angezeigt werden, gesetzt. Namen stehen zwischen "", die Variablen der Formularwerte werden mit einem &-Zeichen hinter die jeweilige Überschrift gesetzt. Alle hier eingegebenen Daten müssen mit einem &-Zeichen verknüpft werden. Bitte achten Sie auf das Leerzeichen vor dem zweiten Anführungszeichen.

```
JMail.Priority = 1
```

Wie oben beschrieben.

```
JMail.Execute
```

JMail wird ausgeführt, und die E-Mail wird verschickt.

Die hier gezeigten Ausführungen sind auch in ähnlicher Weise bei anderen ASP-E-Mail-Skripts wie z. B. ASP Smart Mail oder ASP Mail anwendbar. Es ändern sich hierbei nur die softwarespezifischen Befehle, wie z. B. »JMail.Subject«, es bedeutet bei ASP Mail »Mailer.Subject« usw. Das

Flash-Formular kann so übernommen werden, eventuell ist eine Definierung der einzelnen Flash-Formularfelder in dem ASP-Skript nötig. Dies funktioniert wie oben bei dem Festlegen der Variablen und der Formatierung beschrieben. Da diese Lösungen aber hier nicht ausführlich getestet wurden, kann auch keine ausführliche Beschreibung erfolgen. Kontaktieren Sie in diesem Fall Asp Mail (www.serverobjects.com) bzw. www.Aspsmart.com.

Weitere Informationen zu JMail erhalten Sie unter www.dimac.net.

CGI und Flash

◀
Martin Plücker
www.diconnects.de

Flash bietet mit der `Load Variables`-Anweisung die Möglichkeit, Variablen aus Textdateien auszulesen. Häufig reicht das einfache Lesen jedoch nicht aus. So müssen z. B. bei Gästebüchern auch Daten auf dem Internet-Server gespeichert und später wieder abgerufen werden. Hier kommen nun die CGI-Programme zum Einsatz.

In diesem Kapitel sollen die wichtigsten Techniken, die man in Verbindung mit Flash und CGIs braucht, an mehreren Beispielen demonstriert werden. Am Ende sollen die erworbenen Kenntnisse bei der Entwicklung eines Gästebuches mit Flash und CGI angewandt werden.

Was ist CGI?

Das Common Gateway Interface (CGI) ist eine Möglichkeit, Programme im WWW bereitzustellen, die über das HTTP-Protokoll aus aufgerufen werden und dann Daten in beliebiger Formatierung ausgeben können.

Die Datenverarbeitung geschieht auf dem Internet-Server. CGI-Programme können auf dem Server Daten speichern, zum Beispiel wie oft auf eine WWW-Seite zugegriffen wurde oder was ein Anwender in ein Gästebuch geschrieben hat. Bei entsprechendem Aufruf kann ein CGI-Programm gespeicherte Daten auslesen und daraus z. B. HTML-Code oder beliebige andere Zeichenketten generieren.

Die so genannte CGI-Schnittstelle muss von der WWW-Server-Software unterstützt werden. Wenn man Speicherplatz auf einem WWW-Server mietet und der Server CGIs unterstützt, steht die CGI-Schnittstelle meist in Form eines bestimmten Verzeichnisses zur Verfügung. Meistens hat dieses Verzeichnis den Namen »cgi-bin«. In diesem Verzeichnis können Programme abgelegt werden, welche die CGI-Aufgaben übernehmen.

Es gibt keine Vorschriften dafür, in welcher Programmiersprache ein CGI-Programm geschrieben sein muss. Damit das Programm auf dem Serverrechner ausführbar ist, muss es entweder für die Betriebssystemumgebung des Servers als ausführbares Programm kompiliert worden sein, oder es muss auf dem Server ein Laufzeitinterpreter vorhanden sein, der das Programm ausführt. Wenn der Server z. B. ein Unix-Rechner ist, führt er C-Programme aus, die mit einem Unix C-Compiler zu einer ausführbaren Datei kompiliert wurden. Wenn der Server ein Windows NT-Rechner ist, können CGI-Scripts auch EXE-Dateien sein, die mit 32-Bit-Compilern für C, Pascal, Visual Basic usw. erzeugt wurden. Die meisten heutigen CGI-Programme sind in der Unix Shell-Sprache oder in Perl geschrieben. Die Unix Shell-Sprache wird von allen Unix-Rechnern interpretiert. Für Perl muss ein entsprechender Interpreter installiert sein, den es sowohl für Unix-Systeme als auch für Windows 9x/NT gibt. Dieses Kapitel wird sich mit Perl-CGIs befassen.

Senden von Variablen an CGIs

Damit ein CGI Daten auf einem Server speichern kann, müssen diese von Flash an das CGI gesendet werden. Dies geschieht mit Hilfe der ActionScript-Funktion `Load Variables` (zu finden unter `Load/Unload Movie`). In den Parametern gibt man als URL die Adresse des CGIs an (z. B. http://www.domain.de/cgi-bin/guestbook.pl). Wenn man nun Variablen mit POST/GET versenden einstellt, schickt Flash Variablen an das CGI. Man sollte diese Option auf Variablen mit POST versenden stellen, da bei GET nur eine begrenzte Menge an Daten gesendet werden kann und somit manche Variablen eventuell gar nicht oder nur teilweise mitgesendet werden.

Wenn man `Load Variables` so aufruft, werden alle Variablen, die in dem Film/Symbol, von dem die `Load Variables`-Anweisung aufgerufen wird, gesetzt sind, an das CGI übermittelt.

Beispiel: In dem Filmsymbol »Sender«, das im Hauptfilm liegt, wird mit `Load Variables` ein Gästebuch-CGI aufgerufen. Wenn Variablen mit POST versenden eingestellt ist, werden nun alle Variablen aus dem Sender-Symbol an das Gästebuch-CGI übermittelt.

Wie kann man nur bestimmte Variablen senden?

Da man in den Flash-Filmen häufig auch Steuervariablen benutzt, die nicht an das CGI gesendet werden sollen (z. B. bei dem Gästebuch sollen

nur die Nachricht und der Absender gesendet werden), kann man mit einem Trick nur bestimmte Variablen senden. Dazu erstellt man ein leeres Symbol vom Typ Filmsequenz. Dem Symbol gibt man dann einen beliebigen Instanznamen (z. B. »Server«). In den Aktionen, in denen Variablen an das CGI gesendet werden sollen, setzt man Variablen mit der gleichen Bezeichnung in dem Server-Filmsymbol mit »Set Variables« auf den Wert der Variablen: Wenn man z. B. die Variable `Text` übermitteln will, benutzt man die Aktion `>Set Variables "Server:Text" = Text<`. Auf diese Weise setzt man in dem Symbol alle Variablen, die übermittelt werden sollen.

Anschließend lässt man mit Tell Target das Filmsymbol SERVER die Load Variables-Anweisung ausführen. Es werden dann nur die Variablen, die man in dem Filmsymbol gesetzt hat, gesendet. Damit die Ausgabe des CGIs an den eigentlichen Film gelangt, gibt man bei Load Variables als Position das Ziel »../« an.

Auslesen von Variablen aus CGIs

Das Laden von Variablen, wie z. B. die Einträge aus einem Gästebuch, funktioniert ebenfalls mit der ActionScript-Funktion `Load Variables`. Auch hier gibt man die URL zu dem entsprechenden CGI-Programm an (zum Beispiel http://www.domain.de/cgi-bin/guestbook.pl).

Damit Flash die Variablen aus dem CGI verarbeiten kann, muss das CGI die Variablen als Zeichenkette im Format `variable1=123&variable2=345&variable3=...` ausgeben, die einzelnen `Variablenname=Wert`-Paare werden also durch ein & getrennt. Dabei ist es wichtig, dass die Variablennamen und der Variablenwert selbst keine &-Zeichen beinhalten, da sonst Flash nicht mehr richtig feststellen kann, wo das nächste Paar beginnt. Sollen trotzdem &-Zeichen im Wert von Variablen enthalten sein (z. B. bei Texten), müssen diese vom CGI durch die Zeichenfolge »%26« ersetzt werden.

Die Variablen werden dann in die im Parameter `Position` der `Load Variables`-Anweisung angegebene Filmsequenz geladen.

Auf das Laden der Variablen warten

Nach dem Aufruf der `Load Variables`-Anweisung sind die Variablen, wie beim Laden aus Textdateien, nicht sofort verfügbar – es muss zuerst eine Verbindung zum Server aufgebaut und die Variablen müssen übertragen werden. Auch beim Senden sollte man warten, bis der Server die Daten

empfangen hat. Aus diesem Grund muss man Flash dazu bringen, so lange zu warten, bis die Variablen vollständig übertragen wurden.

Zu diesem Zweck lässt man das CGI am Ende der Zeichenkette noch eine zusätzliche Variable (z. B. `result`) anhängen, die das Ende der Zeichenkette markiert und angibt, ob das CGI seine Funktion erfolgreich ausführen konnte. An dieser Variablen kann man dann feststellen, wann alles übertragen wurde und ob die Aktion erfolgreich war.

Vor dem Aufruf der `Load Variables`-Anweisung wird diese End-Variable mit `Set Variables` auf den Wert –1 gesetzt. Nach dem Aufruf von `Load Variables` erstellt man dann zwei Schlüsselbilder. Dem ersten gibt man eine beliebige Bezeichnung. In den Aktionen des ersten Schlüsselbildes fragt man ab, ob der Wert der End-Variablen sich verändert hat. Wenn dem nicht so ist, überprüft man den Wert der End-Variablen. Wenn die Aktion ein Erfolg war, kann man den Film mit `Goto` an einer anderen Stelle normal weiterlaufen lassen oder einfach stoppen. Bei einem Fehler kann man diesen dem User mitteilen.

Es hat sich als günstig erwiesen, der End-Variablen im Erfolgsfall den Wert 0 zu geben. Wenn bei der Ausführung des CGIs ein Fehler auftritt, sollte dieses in der End-Variablen als Wert einen Fehlercode > 0 ausgeben. Die vom CGI ausgegebene Zeichenkette sähe im Erfolgsfall also so aus:

```
"variable1=123&variable2=345&variable3=678&result=0".
```

CGIs mit verschiedenen Funktionen aufrufen

Häufig müssen CGIs mehrere Aufgaben erfüllen (z. B. beim Gästebuch Einträge auf dem Server speichern und später wieder laden), und meist möchte man diese Funktionen von einem einzelnen CGI ausführen lassen. Dann muss man dem CGI beim Aufrufen mit »Load Variables« allerdings mitteilen, welche Funktion ausgeführt werden soll, ob es nun z. B. schreiben oder lesen soll. Dies geschieht, indem man dem CGI beim Aufruf eine Variable mit sendet (siehe »Senden von Variablen an CGIs«), die angibt, welche Funktion es nun ausführen soll. Häufig wird dabei die Variable `action` verwendet.

Entwicklung der CGIs

Über die Entwicklung von Perl-CGIs kann man ganze Bücher füllen. Deshalb sei hier auf die entsprechende Fachliteratur verwiesen.

Perl ähnelt im Syntax sehr JavaScript oder C. Falls Sie bereits über Programmiererfahrungen verfügen, ist die Dokumentation der Perl-Sprachelemente in Self-HTML (http://www.teamone.de/selfaktuell/) ein guter Einstieg. Dort werden die wichtigsten Befehle von Perl anschaulich mit Beispielen erklärt.

Eine weitere Möglichkeit ist, CGIs für HTML-Seiten so zu verändern, dass man sie für Flash einsetzen kann. Dazu muss lediglich die Ausgabe des CGIs auf das für Flash verständliche Format geändert werden.

Installation von CGIs auf Servern

Um ein CGI auf einem Internet-Server zu installieren, müssen Sie auf Ihrem Server über ein Verzeichnis verfügen, in dem Sie eigene CGIs ausführen können. Meist heißt dieses Verzeichnis »cgi-bin«.

Bei allen Perl-CGIs steht in der ersten Zeile nach einem »#!« der Pfad zum Perl-Interpreter auf dem Server. Dieser ist normalerweise »/usr/bin/perl«, er kann aber von Server zu Server verschieden sein. Fragen Sie Ihren Provider danach, oder lesen Sie die Serverdokumentation Ihres Providers.

Neben dem Perl-Pfad sind häufig noch weitere Konfigurationen in den CGIs (z. B. andere Pfade oder Verhaltensweisen) nötig. Lesen Sie dazu die Dokumentation des CGIs.

Um das CGI auf dem Server zu installieren, laden Sie es mit einem FTP-Programm auf den Server. Beachten Sie dabei unbedingt, dass der Übertragungsmodus auf TEXT eingestellt ist.

Bei Unix- oder Linux-Servern müssen Sie anschließend die Zugriffsrechte für das CGI so setzen, dass alle Lese- und Ausführungsrecht und nur der Eigentümer Schreibrecht hat. Dies können Sie entweder direkt mit Ihrer FTP-Software oder über Telnet mit dem Befehl `chmod 755 cgi-name.pl` erledigen. Sollte das CGI Dateien auf den Server schreiben, müssen bei Unix-/Linux-Servern auch dort die Zugriffsrechte auf die Dateien auf Schreibrecht für alle eingestellt werden (per FTP oder mit Telnet `chmod 777 dateiname`).

Anschließend sollte das CGI wie gewollt seinen Dienst verrichten.

Gästebuch mit Flash und Perl-CGI

Hier sollen Sie nun die oben erklärten Techniken an der Entwicklung eines Gästebuchs in Flash anwenden. Um Ihnen die Arbeit zu erleichtern, hat die DiCONNECTS GmbH ein fertiges Gästebuch-CGI zur Verfügung gestellt.

Das »guestbook.pl«-CGI

Das CGI »guestbook.pl« soll die Gästebucheinträge auf dem Server speichern und lesen können. Dazu werden die zwei Funktionen »read« und »write« benötigt. Die Einträge werden auf dem Server in die Datei guestbook.txt gespeichert.

Welche Funktion aufgerufen werden soll, wird durch das Senden der Variablen `action` bestimmt.

Wenn ein Eintrag gespeichert werden soll (`"action" = "write"`), müssen zusätzlich noch die Variablen `name`, `text`, `email` und `url`, die den Inhalt des Eintrags beinhalten, gesendet werden. Wenn die Variablen `name` und `text` nicht gesendet werden oder leer sind, gibt das CGI einen Fehlercode an Flash zurück, `email` und `url` sind dagegen optional. Beim Speichern fügt das CGI dem Eintrag automatisch das aktuelle Datum hinzu.

Für das Lesen der Einträge (`"action" = "read"`) müssen keine weiteren Variablen gesendet werden. Das CGI gibt die Einträge dann durchnummeriert aus, die Nummerierung beginnt bei 0. Zusätzlich zu den eigentlichen Einträgen liefert das CGI auch noch die Anzahl der Einträge in der Variablen `count`. Die zurückgegebene Zeichenkette sieht im Erfolgsfall z. B. so aus:

```
"name0=Hans&text0=Hallo&email0=post@hans.de&url0=http://
www.hans.de&date0=10.10.2000
10:31&name1=...&text1=...&...&count=2&result=0"
```

Im Fehlerfall wird nur die End-Variable mit einem Fehlercode zurückgegeben:

1. Die Datei guestbook.txt konnte nicht gefunden werden oder hat nicht die richtigen Zugriffsrechte
2. Es wurde die Variable `name` nicht übermittelt, oder sie ist leer.
3. Es wurde die Variable `text` nicht übermittelt, oder sie ist leer.

Konfiguration und Installation des »guestbook.pl«-CGIs

Kopieren Sie die Datei guestbook.txt auf Ihren Server. Wenn der Server ein Unix- oder Linux-System ist, müssen Sie die Zugriffsrechte der Datei auf Lese- und Schreibrecht für alle setzen (über FTP oder Telnet mit `chmod 666 guestbook.txt`).

Öffnen Sie nun die Datei guestbook.pl in einem Texteditor. Passen Sie gegebenenfalls den Pfad zum Perl-Interpreter in der ersten Zeile an. Unter dem Copyright-Header finden Sie die Zeile `>$basedir='/home/xyz';<`. Ersetzen Sie `/home/xyz` durch das Verzeichnis, in das Sie die guestbook.txt kopiert haben (die einfachen Hochkommata stehen lassen). Etwas weiter unten finden Sie die Zeile `>$useflock=0;<`. Wenn Sie auf einem Unix- oder Linux-Server arbeiten, tragen Sie hier statt 0 eine 1 ein. Speichern Sie das CGI.

Kopieren Sie das CGI danach auf Ihren Internet-Server in Ihr CGI-Verzeichnis (meist »cgi-bin«, achten Sie auf den Text-Übertragungsmodus). Bei einem Unix- oder Linux-Server müssen Sie wieder die Zugriffsrechte auf Lese- und Ausführungsrecht für alle und Schreibrecht für den Eigentümer setzen (über FTP oder Telnet mit `chmod 755 guestbook.pl`).

Das CGI ist nun einsatzbereit.

Erstellung der Flash-Oberfläche

Erstellen Sie nun eine Oberfläche für ein Gästebuch, die mit dem CGI zusammenarbeitet. Achten Sie darauf, die im ersten Teil erklärten Techniken zu verwenden. Die Gästebuch-Oberfläche soll Folgendes leisten:

- Gästebucheinträge vom CGI laden und auf das Laden warten
- Gästebucheinträge anzeigen
- Möglichkeit zum Schreiben eines neuen Eintrages bieten
- Neuen Eintrag an das CGI senden und auf eine Erfolgsmeldung warten
- Eventuell auftretende Fehler erkennen und anzeigen

Eine voll funktionsfähige Musterlösung finden Sie in dem guestbook.fla. Darin finden Sie in allen ActionScripts Kommentare, welche die Funktionsweise erläutern. Um die Musterlösung zu testen, setzen Sie die Variable `cgiurl` in den Aktionen des ersten Frames (»Config«) auf die URL zu Ihrem »guestbook.pl«-CGI.

Textdateien editieren

▶ Mario Mies
www.flashbeat.de

Das folgende Tutorial erläutert, wie Sie mit Flash 4 und 5 und zwei einfachen CGI-Skripts, die bei Flashern so beliebten Textdateien online editieren bzw. den Inhalt löschen können. Somit können Sie auch Ihren Kunden die Möglichkeit bieten, relativ einfach die Texte auf den jeweiligen Seiten zu editieren.

In dem Skript ist eine Passwortabfrage integriert, die auch die Einlog-Versuche speichern kann und somit Änderungen nachvollziehbar macht. Es muss also sichergestellt sein, dass Ihr Webspace Account eigene CGI-Skripts und einen Perl-Interpreter unterstützt. Die eigentlichen Textdateien sollten der Übersicht halber in einem separaten Verzeichnis gespeichert werden (bei diesem Beispiel in dem Verzeichnis Text), das vor direkten Lese- und Schreibzugriffen geschützt ist. Die .swf-Datei inklusive der dazugehörigen .html-Datei sowie die Dateien data.txt und log.txt sollten auch in diesem Verzeichnis liegen. Die Datei passw.html, die auch als Startdatei für das Formular dient, enthält das Passwort Formular, die Datei sollte im Hauptverzeichnis liegen.

In dem hier vorgestellten Flash-Film wird ein Scrollbalken für das Textfeld benutzt; wie diese Scrollbalken erstellt werden, können Sie in einem separaten Tutorial nachlesen.

Zum ersten Erstellen der Textdateien benutzen Sie am besten einen simplen Texteditor, zum Hochladen können Sie handelsübliche FTP-Programme verwenden, es ist dabei aber nicht notwendig, die Dateien im ASCII-Format zu übertragen.

Zur Flash-Programmierung

Erstellen Sie als Erstes ein Texteingabefeld mit Namen: »file«. In diesem Eingabefeld geben Sie relativ zum CGI-Verzeichnis (../Verzeichnis) das (die) Verzeichnis(se) an, in dem die Textdateien liegen. Dies ist notwendig, damit der Flash-Film die Textdaten einlesen kann. Hinter dem letzten Slash kann dann der jeweilige Dateiname online eingegeben werden. Als Zweites erstellen Sie ein weiteres Texteingabefeld mit Namen: »Text«, dieses dient dazu, den Text der Dateien einzulesen bzw. neuen Text einzugeben. Fügen Sie jetzt noch den schon oben erwähnten Scrollbalken hinzu. Im nächsten Schritt müssen die Menüpunkte festgelegt werden. Dabei ist es gleich, ob die Menübuttons als Text oder grafische Buttons angelegt werden. Nachfolgend sehen Sie die einzelnen Aktionen, die jedem Button zugeordnet werden.

Erster Button: neuen Text eingeben

```
On (Release)
   Set Variable: "Text" = "Text="
End On
```

Hier wird dem Texteingabefeld die Flash-Formatierung zugewiesen, damit die jeweilige Flash-Seite den Text aus der Textdatei lesen kann; für jede Textvariable in der Textdatei ist eine Formatierung mit der Syntax `"Name="` anzulegen.

Zweiter Button: Inhalt anzeigen

```
On (Release)
   Load Variables (file, "")
      Begin Tell Target ("/editmc")
   Play
   End Tell Target
End On
```

Hier wird der Text der Datei geladen, deren Name in das Texteingabefeld eingetragen wurde. Achten Sie darauf, die Variable auf Expression zu stellen, des Weiteren wird ein MovieClip gestartet, der den dritten Menüpunkt verdeckt.

Dritter Button: aktuellen Text editieren

```
On (Release)
   Set Variable: "Text" = "Text=" & Text
End On
```

Hier wird dem Texteingabefeld die Flash-Formatierung zugewiesen, damit die jeweilige Flash-Seite den Text aus der Datei lesen kann. Für jede Textvariable ist eine Formatierung mit der Syntax `Name=` anzulegen. Die Formatierung wird in diesem Falle vor den Text gesetzt, damit wird der bereits vorhandene Text in der jeweiligen Textdatei überschrieben.

Vierter Button: Inhalt löschen

```
On (Release)
   Set Variable: "file" = file
   Get URL ("loesch.pl", vars=POST)
End On
```

Hier wird eine Variable festgelegt, die dann mit dem GetURL-Befehl an das CGI-Skript übergeben wird. Mit `file` wird der Pfad des ersten Textfeldes eingelesen. Der gesamte Inhalt der Textdatei wird gelöscht. Wenn Sie die Passwortabfrage benutzen, wird die Seite in einem CGI-Skript aufgerufen. Somit müssen auch die Pfade darauf abgestimmt sein, d.h relativ zum CGI-bin-Verzeichnis

Fünfter Button: Zur Datei hinzufügen

```
On (Release)
   Set Variable: "Text" = Text
   Set Variable: "file" = file
   Get URL ("hinzu.pl", vars=POST)
End On
```

Hier werden Variablen festgelegt, die dann mit dem GetURL-Befehl an das CGI-Skript übergeben werden, mit `file` bzw. `Text` wird der Inhalt der Textfelder eingelesen. Der gesamte Inhalt des Textfeldes wird der Textdatei hinzugefügt. Wenn Sie die Passwortabfrage benutzen, wird die Seite in einem CGI-Skript aufgerufen, somit müssen auch die Pfade darauf abgestimmt sein, d.h. relativ zum CGI-bin-Verzeichnis.

Zu den CGI-Skripts

Die beiden Perl-Scripts dienen dazu, die Daten zu löschen bzw. den Text den einzelnen Dateien hinzuzufügen. Die Handhabung ist sehr einfach, da die Variablen von dem Flash-Film gesteuert werden. Die einzige Änderung in dem Skript (sofern Sie die Namen der Texteingabefelder aus unserem Skript übernehmen) betrifft den Pfad zu dem Passwortformular. Dies ist in den Skripts `hinzu.pl` und `loesch.pl` die jeweils letzte Zeile. Die Passwortdatei wird weiter unten behandelt.

Bitte achten Sie bei Linux-Providern darauf, den `chmod` der .pl-Dateien im CGI-Verzeichnis auf 755 zu setzen. Je nach Konfiguration des Servers

ist auch ein chmod 777 für die Textdateien bzw. das Formular nötig. Achten Sie bei NT-Providern darauf, die Textdateien in einem Verzeichnis zu speichern, das indirekte Lese- und Schreibrechte bietet, da hier kein `chmod` gesetzt werden kann.

Ein wichtiger Punkt betrifft die Version des Perl-Interpreters Ihres Providers, sie muss mindestens der Version 5.004 entsprechen, bzw. bei älteren Versionen muss das Modul `cgi.pm` eingerichtet sein. Um die Interpreter-Version und weitere Informationen zu dem verwendeten Server zu ermitteln, liegt der Buch-CD ein Skript mit Namen tester.cgi bei. Einfach in das CGI-Verzeichnis kopieren und im Browser aufrufen. Die dabei erzeugte Seite zeigt einige technische Merkmale des von Ihnen verwendeten Servers.

Zur Passwortabfrage

In der Datei Passw.html liegt das Formular zum Eingeben des Benutzernamens und des Passwortes. Als Action-Einstellung geben Sie den Pfad zu der pwd.pl-Datei an (bei unserem Beispiel: cgi-bin/pwd.pl). Auf dem Server wird diese Datei aufgerufen, und die dort gemachten Einstellungen werden ausgeführt. Bitte achten Sie darauf, in der .html-Datei, die den Flash-Film enthält (bei uns txtform.html), den Pfad zu dem .swf-File relativ zu verändern (z. B. ../Text/txtform.swf oder je nach Provider auch absolut z. B. www.url.de/Text/txtform.swf).

Folgende Einstellungen sind in der Datei pwd.pl zu machen:

```
$file='../Text/txtform.html';
```

Pfad zu der Datei, die geschützt werden soll.

```
$pwdfile='../Text/data.txt';
```

Pfad zu Passwortdatei, die Syntax für die Zugangsdaten lautet: Benutzername|Passwort, bitte keine Leerzeichen eingeben.

```
$logit="yes";
```

Sollen die Einlog-Versuche gespeichert werden mit »yes« oder »no«.

```
$logfile='../Text/Text.txt';
```

Pfad zu der Datei, welche die Einlog-Versuche speichert.

```
$yourname='Max Mustermann';
```

Name des Webmasters, der im Problemfall kontaktiert werden soll.

```
$youremail='max.mustermann@gmx.de';
```

E-Mail des Webmasters, der im Problemfall kontaktiert werden soll.

```
$posthere = "pwd.pl";
```

URL zum diesem Skript, relativ zum CGI-Verzeichnis.

```
data.txt
```

Die Passwortdatei im ASCII-Modus hochladen.

```
log.txt
```

Die Log File-Datei im Binary-Modus hochladen.

Flash und XML

Im Folgenden erläutern wir Ihnen kurz, was XML ist und wie man seine Vorteile in Zusammenarbeit mit Flash nutzen kann.

XML-Einführung

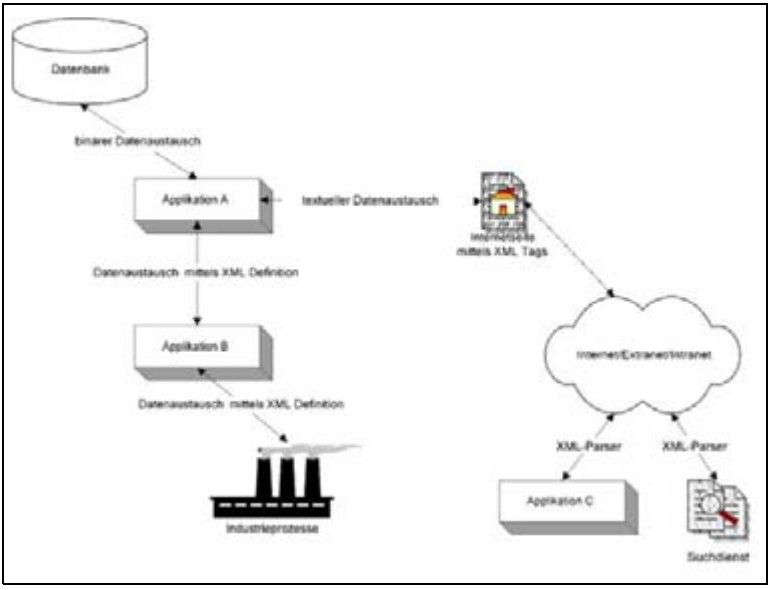
▸ Christoph Aigner
www.alaris.at

XML, die Abkürzung für Extensible Markup Language, ist ein strukturiertes Format, um Daten über Internettechnologien auszutauschen. Der Begriff Daten umfasst sowohl Content-Daten von Applikationen als auch Steuerungsdaten von Prozessen. Applikationen, wie beispielsweise Enterprise Ressource Planning-Systeme (ERP), oder Kalkulationsprogramme speichern Ihre Daten entweder im binären Format (in der Datenbank) oder im textuellen Format (in der Regel für den Datenimport und -export).

XML verwendet letzteres Format und bietet die Möglichkeit, plattformunabhängig und standardisiert Schnittstellen zu beschreiben, um den Datenaustausch zu ermöglichen. Diese Schnittstellendefinitionen können sowohl direkt zum Datenaustausch zwischen Applikationen verwendet werden und können auch Internetseiten als Daten-Gateway benutzen. Beides findet heute bereits Anwendung, B2B-Systeme verwenden XML-Connectoren für den Datenaustausch zwischen den einzelnen Applikationen, während Content Management-Systeme Internetseiten mittels XML strukturieren, um Suchdiensten und Web-Crawlern die Möglichkeit zur einfachen Strukturierung zu bieten. Abbildung 1 soll die Verwendung von XML illustrieren.

Wo liegt nun der Vorteil von XML? In Wahrheit sind XML-Dateien doch Textdateien, die es bereits gibt und jeder kennt (HTML, CSV oder durch Tabulatoren getrennte ASCII-Dateien)? XML bietet nun den Vorteil, dass die Struktur dieser textuellen Informationen, die wir heutzutage haben, standardisiert ist und es so für alle Applikationen möglich ist, diese zu lesen.

XML verwendet für die Beschreibung der Datenstruktur eine ähnliche Syntax wie HTML. Jedoch ist diese Syntax von XML wesentlich strikter als die von HTML, da der standardisierte Datenaustausch im Mittelpunkt



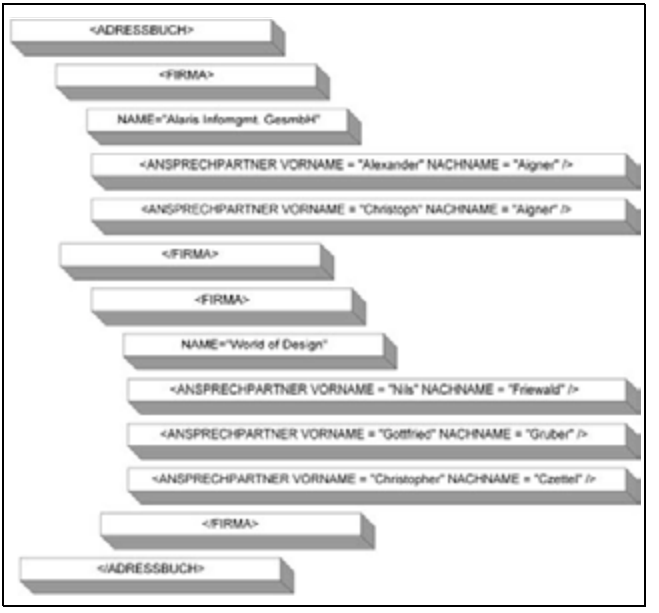
◀ **Abbildung 1**
Anwendungsgebiete von XML

steht. Obwohl wir immer von XML als solches sprechen, beschreibt der Begriff eine Familie von Technologien. Der XML 1.0-Standard enthält die Beschreibung, wie mittels Tags und Attributen Information zu strukturieren ist. Darüber hinaus gibt es aber noch weitere Definitionen, die den Standard erweitern. Xlink beispielsweise ist eine Definition, um Linkstrukturen zu beschreiben. XSL ist ein Standard, der Stylesheets definiert. Der Document Object Model-(DOM-)Standard wird verwendet, um Datenstrukturen zwischen Applikationen auszutauschen. Die Technologiefamilie XML wird ständig erweitert, um den Anforderungen der heutigen Informationsgesellschaft gerecht zu werden.

Warum XML in Flash?

XML bietet die Möglichkeit, professionelle Datenschnittstellen bereitzustellen. Durch die Möglichkeit, mehrere Werte zu einem Knoten zusammenzuschließen und somit durch eine logische Struktur abzubilden, können Daten effizient und verständlich übermittelt werden. Der in vorigen Versionen von Flash zur Anwendung gekommene Weg, Variablen und Werte getrennt durch &-Zeichen auszutauschen, ergibt eine unübersicht-

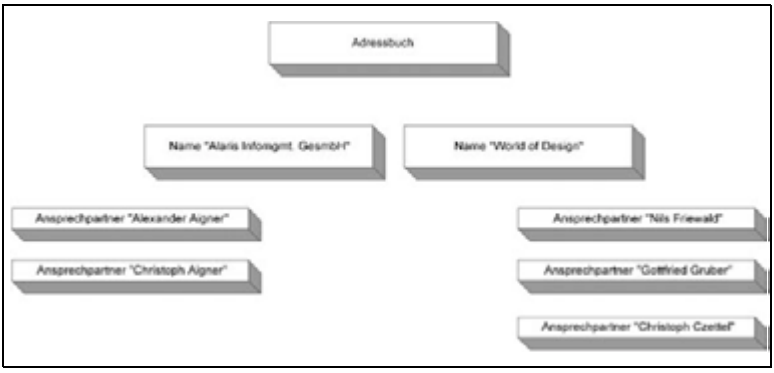
Abbildung 2 ▶
XML-Tags,
Beispiel Adressbuch



liche Variablenkette (z. B.: PERSON1_Vorname=Max&PERSON1_Nachname=Mustermann&PERSON_2_Vorname=Eduard&PERSON2_Nachname=Mars ...).

Sobald einmal längere und kompliziertere Datenstrukturen übergeben werden müssen, wird das Ganze unüberschaubar. Je größer das Projekt und je mehr Personen an einem Projekt arbeiten, desto unübersichtlicher und fehleranfälliger wird der ganze Informationsfluss.

Die nachstehend erklärte DOM-Struktur von XML bietet die Möglichkeit, Informationen in Hierarchien darzustellen und Ebenen festzulegen. Dadurch wird der Informationsblock übersichtlich und transparent gehalten. Bei größeren Projekten (vielleicht auch in Zusammenhang mit anderen Programmen) lernt man die Vorteile von XML kennen, auch wenn es auf den ersten Blick ziemlich aufgepuscht wirkt. Vom Denkansatz wird eine Form der Codeübermittlung beschrieben (eine Struktur) und diese dann mit Daten und Werten gefüllt. Beim normalen Variablenlisting (der einzige Weg vor Flash 5) werden Werte ohne Konzeption, Struktur oder Kontrollmechanismen aneinander gehängt und dieser flache Block an Daten an die Zielapplikation geschickt. Bei kurzen Informationen sicher eine bequeme Art der Übertragung, es gibt jedoch Einschränkungen beim Ausbau bzw. bei Erweiterungen.



◀ Abbildung 3
Informationsstruktur Adressbuch

Document Object Model (DOM)

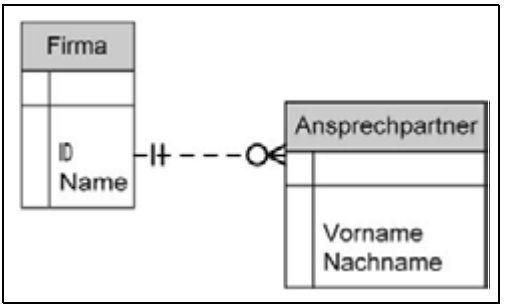
Prinzipiell können XML-Dokumente beliebig strukturiert werden. Das DOM bietet allerdings einen standardisierten baumorientierten Aufbau, um Informationen zwischen Applikationen unabhängig von der verwendeten Programmiersprache und der Plattform auszutauschen. Der Vorteil von DOM ist die Verfügbarkeit von Parsern in vielen Programmiersprachen (Java, C++, Perl, Python, Flash 5 etc.). Dadurch ist es besonders einfach, Datenschnittstellen auf der DOM-Struktur zu realisieren. Das DOM liegt mittlerweile in einer überarbeiteten Version 2 vor und wird vom W3C als Standard empfohlen. Der Unterschied zur früheren 1.0-Version liegt vor allem in der Erweiterung von Funktionsparametern und dem Exception-Handling (= Fehler- und Störungsbehandlung).

DOM baut eine Information objektorientiert und baumartig auf. Abbildung 2 soll dies verdeutlichen.

Die Abbildung zeigt die XML-Tag-Struktur eines Adressbuches. Alle Entitäten sind über eine Baumstruktur ineinander verschachtelt. Auf logischer Ebene entspricht dieses XML-Beispiel der Informationsstruktur in Abbildung 3.

Obiges Adressbuchbeispiel wird von einer Applikation in der Regel in einer Datenbank binär abgespeichert. Da die meisten Daten heutzutage nicht objektorientiert, sondern sequenziell über SQL-Befehle gesteuert werden, müssen XML-Strukturen vor Abspeicherung umgewandelt werden. Eine Tabellenstruktur zu unserem Beispiel könnte daher wie in Abbildung 4 dargestellt aussehen.

Abbildung 4 ▶
Crow's Foot Tabellendiagramm Adressbuchbeispiel



Um die XML-Struktur des Adressbuchbeispiels mit Flash 5 umzusetzen, wird folgender Code benötigt:

```
XMLpage = new XML();

XMLadressbuch = XMLpage.createElement("Adressbuch");

XMLfirma = XMLpage.createElement("Firma");

XMLfirma.attributes.Name = "Alaris, Informationsmanagement GmbH";
XMLansprechpartner = XMLpage.createElement("Ansprechpartner");
XMLansprechpartner.attributes.Vorname = "Christoph";
XMLansprechpartner.attributes.Nachname = "Aigner";
XMLfirma.appendChild(XMLansprechpartner);

XMLansprechpartner = XMLpage.createElement("Ansprechpartner");
XMLansprechpartner.attributes.Vorname = "Alexander";
XMLansprechpartner.attributes.Nachname = "Aigner";
XMLfirma.appendChild(XMLansprechpartner);

XMLadressbuch.appendChild(XMLfirma);

XMLfirma = XMLpage.createElement("Firma");

XMLfirma.attributes.Name = "World of Design";
XMLansprechpartner = XMLpage.createElement("Ansprechpartner");
XMLansprechpartner.attributes.Vorname = "Nils";
```

```
XMLansprechpartner.attributes.Nachname = "Friewald";
XMLfirma.appendChild(XMLansprechpartner);

XMLansprechpartner = XMLpage.createElement("Ansprechpartner");
XMLansprechpartner.attributes.Vorname = "Gottfried";
XMLansprechpartner.attributes.Nachname = "Gruber";
XMLfirma.appendChild(XMLansprechpartner);

XMLansprechpartner = XMLpage.createElement("Ansprechpartner");
XMLansprechpartner.attributes.Vorname = "Christopher";
XMLansprechpartner.attributes.Nachname = "Czettel";
XMLfirma.appendChild(XMLansprechpartner);

XMLadressbuch.appendChild(XMLfirma);

XMLpage.appendChild(XMLadressbuch);

trace("XML: " + XMLpage.toString()); //Gibt den String im Outputwin-
dow aus
```

Durch den Aufruf von `XMLpage = new XML();` wird ein neues XML-Objekt erzeugt. Diesem XML-Objekt wird durch den Aufruf `XMLansprechpartner = XMLpage.createElement("xy");` durch ein Element (`childnode`) erweitert.

Dem soeben generierten Element werden durch die Zuweisung von Attributen die individuellen Parameter zugewiesen. Dies geschieht durch den Aufruf: `XMLansprechpartner.attributes.Vorname = "Christoph";`.

Im Anschluss (nach der Definition der Attribute) wird das Element mittels `XMLadressbuch.appendChild(XMLansprechpartner);` am XML-Dokument an den Knoten Adressbuch angehängt.

Sobald auf diese Weise das Dokument fertig definiert worden ist, gibt es zwei unterschiedliche Verfahren:

- Dauerhafte Verbindung zum Server
  Es muss vorher eine Verbindung zum Server aufgebaut werden (`socket.connect`). Danach wird durch den Eintrag `socket.onXML = XMLinput;` eine Callback-Funktion eingebunden. Diese Funktion wird aufgerufen, sobald Daten vom Server empfangen werden. Also muss die Funktion `XMLinput(doc)` zuvor deklariert worden sein. In `doc` ist dann das empfangene XML-Dokument enthalten. Die Verbindung bleibt bestehen, bis entweder der Server die Verbindung beendet, die

Leitung durch höhere Gewalt zusammenbricht, der User das Flash-Movie stoppt oder der Befehl `socket.close();` aufgerufen wird.

- Ansteuerung eines serverseitigen Skriptes (PHP, ASP, CGI ...)
  Für diese Art der Informationsübermittlung zwischen Flash und einem Server-Skript muss ein Dokument generiert werden, das die vom Server kommenden Daten beinhaltet. Dies geschieht durch den Aufruf von `XMLreturn = new XML();`. Weiter muss wiederum die Callback-Funktion genannt werden, die das vom Server zurückgegebene Datenpaket empfängt. Dies wird durch die Funktion `XMLreturn.onLoad = onXMLReply;` erzielt.

Parsen des XML-Dokuments

Das vom Server erhaltene Dokument kann man durch die Aufrufe von `childNodes` und `attributes` in Variablen zurückverwandeln.

Beispiel:

```
Nodes = doc.childNodes;
For (i = 0; i < Nodes.length; i++)  {
   If (Nodes[i].nodeName == "Ansprechpartner")
       {
         trace("VORNAME: " + Nodes[i].attributes.Vorname);
         trace("VORNAME: " + Nodes[i].attributes.Nachname);
       }
}
```

Falls mehrere Knoten verschachtelt sind, könnte man das Ganze in eine Rekursion packen und die Funktion so oft aufrufen, bis `Nodes.haschildNodes() == false` ist.

Flash-Newsletter

Hier sehen Sie ein weiteres Beispiel für aus Flash angesprochene Skripte. Sie erhalten in diesem Kapitel ein Skript zur Administrierung eines Newsletters.

▸ Marco Asbach
www.partylogger.de

Was kann der Newsletter?

1. Ein- und Austragen von Abonnenten über Ihre Webseite
2. Prüfung auf Eingabe unkorrekter E-Mail-Adressen
3. Beim Eintragen Überprüfung auf doppelte E-Mail-Adressen
4. Beim Austragen Überprüfung darauf, ob die Adresse überhaupt eingetragen ist
5. Fehlermeldung, falls das CGI-Skript einmal nicht verfügbar sein sollte.

Was ist vor dem Upload zu tun?

Folgende Dateien sind in der Zip-Datei enthalten:

- news.pl: Das CGI-Skript zur Ein- und Austragung von Abonnenten Ihres Newsletters.
- cfg.pl: Das Konfigurationsskript.
- mail-admin.pl: Das Administrierungstool. Mit diesem können Sie die Einträge selbst pflegen oder den Newsletter erstellen und versenden.
- pass.dat: In dieser Datei wird Ihr Passwort abgespeichert.
- n_letter.fla: Die Flash-Datei zur Erstellung Ihres Movies.
- list.txt: In dieser Datei werden die E-Mail-Adressen gespeichert.
- footer.txt: Enthält die Fußzeile Ihres Newsletters.
- header.txt: Enthält die Kopfzeile Ihres Newsletters.

Anpassen der CGI-Skripts

In den Dateien mail-admin.pl und news.pl finden Sie in der ersten Zeile den Eintrag:

```
#!/usr/bin/perl
```

Dies ist die Pfadangabe für Ihren Perl-Interpreter. Wenn Sie nicht genau wissen, wie Ihr Pfad lautet, nehmen Sie mit Ihrem Provider Kontakt auf oder schauen auf den Support-Seiten Ihres Providers nach. Passen Sie diesen an.

In der Datei cfg.pl finden Sie folgenden Eintrag:

```
$mailprog="/usr/bin/sendmail -t";
```

Dies ist die Pfadangabe zu dem sendmail-Programm Ihres Servers. Auch hier gilt: Sprechen Sie im Zweifel Ihren Provider an. Passen Sie auch diesen Pfad an Ihren an. Ferner gibt es in dieser Datei den Eintrag:

```
$main_page="http://www.IhreDomain.de";
```

Tragen Sie hier Ihren Domainnamen ein. Und die letzte Änderung betrifft die Zeile:

```
$yourmail="email\@email.de";
```

Hier legen Sie die Adresse fest, die als Absender des Newsletters erscheint bzw. die für E-Mail-Meldungen an Sie verwendet wird. Wichtig ist, dass Sie den Backslash \ nicht vergessen.

Anpassen der Kopf- und Fußzeile Ihres Newsletters

Öffnen Sie jeweils die Dateien header.txt und footer.txt und passen die Texte an Ihre Wünsche an.

Anpassen der FLA-Datei

Öffnen Sie die FLA-Datei in Flash und passen das Layout an Ihre Vorgaben an. Achtung: Es gibt einiges an ActionScript in dieser Datei. Seien Sie vorsichtig, sonst funktioniert Ihr Newsletter nachher nicht einwandfrei. Wichtig ist, dass Sie die Action `Load Variables` des ABSENDEN-Buttons im ersten Frame anpassen. Geben Sie hier die relative URL an, an der Sie das CGI-Script news.pl auf Ihrem Server abgelegt haben.

Datei-Upload

So, das war es an Vorbereitung. Nun können Sie die Dateien laden. Die Dateien cfg.pl, news.pl, list.txt, header.txt, footer.txt, pass.dat und mail-admin.pl werden alle in das CGI-Verzeichnis Ihres Servers aufgespielt. Achtung: Überspielen Sie diese Dateien alle im ASCII-Modus. Schauen Sie dazu in die Bedienungshilfe Ihres FTP-Programmes. Ihr *.swf spielen Sie am besten ein Verzeichnis höher ein. Dieses wird aber wiederum im Binary-Modus auf den Server übertragen.

Benutzerrechte festlegen

Nun müssen Sie die Nutzerrechte der Dateien auf Ihrem Server festlegen, den so genannten `chmod`. Wie Sie dies machen, entnehmen Sie am besten der Bedienungshilfe Ihres FTP-Programmes. Folgende Einstellungen sind vorzunehmen:

```
cfg.pl chmod 755
news.pl chmod 755
mail-admin.pl chmod 755
alle txt- und dat-Dateien werden auf chmod 777 gesetzt
```

Administrierung

Testen können Sie die Funktionsweise in der Regel nur online. Wenn Sie aber alles beachtet haben, sollte Ihr Newsletter nun funktionieren. Zur Administration geben Sie als URL direkt die URL auf mail-admin.pl an (z. B.: http://www.meinprovider.de/cgi-bin/mail-admin.pl). Beim ersten Zugriff auf dieses Skript werden Sie zur Eingabe eines Passwortes aufge-

fordert. Wenn Sie dies getan haben, hilft ein Reload oder ein Schließen und neues Öffnen des Browsers weiter. Nun können Sie immer über die URL des mail-admin.pl (nach Eingabe Ihres Passwortes) einen Newsletter versenden oder die Adressliste pflegen.

Ming – SWFs ohne Flash

Ming, die SWF-Ausgabebibliothek

Ming ist eine C-Bibliothek, um SWF-Dateien zu erzeugen, sowie ein Set von Wrappern für gängige Skriptsprachen wie PHP, Perl oder Phyton. Fast alle grafischen Funktionen von Flash 4 werden bereits in der aktuellen Version v0.04 unterstützt.

▶ Wojciech Kalka
www.kalka.org

Für dynamische Flash-Webseiten ist es schon lange notwendig, Server Side Flash-Dateien zu erzeugen. Im Vergleich zu Programmen wie Macromedia Generator oder dem Swish Generator geht die Ming-Bibliothek von einem grundlegend anderen Ansatz aus. Ohne jegliche Kenntnis und Nutzung von Flash lassen sich Grafiken und Animationen im Flash SWF-Dateiformat erzeugen. Ming bietet für Programmierer ein Hilfsmittel, das Ihnen endlich die Möglichkeit gibt, in SWF-Dateien direkt zu programmieren.

Was braucht man, um Ming effektiv zu nutzen? Zuerst sollte ein Webserver mit einer Skriptsprache zur Verfügung stehen. In unserem Testsystem kam der Webserver Apache und die Programmiersprache PHP 4 in der Version 4.02 zum Einsatz. Als Betriebssystem diente Debian Linux 2.2.

Diese Umgebung bildet schon oft die erste Hürde, um Ming zu benutzen. Hat man jedoch ein solches System lauffähig, wird man durch viel Leistung von Ming vollständig entschädigt. Obwohl sich das Projekt, das unter der LGP-Lizenz entwickelt wird, noch in einem recht frühen Stadium befinden, ist der Funktionsumfang der Bibliothek schon sehr umfangreich.

Hello World mit Ming:

```
<?   $f = new SWFFont("Techno.fdb");
     $t = new SWFText();
     $t->setFont($f);
     $t->moveTo(200, 2400);
```

```
$t->setColor(0xff, 0xff, 0);
$t->setHeight(1200);
$t->addString("Hello World");

$m = new SWFMovie();
$m->setDimension(5400, 3600);

$m->add($t);

header('Content-type: application/x-shockwave-flash');
$m->output();  ?>
```

An diesem einfachen Beispiel kann man schon erahnen, welche Möglichkeiten sich hier einem erfahrenen Benutzer eröffnen. So lassen sich in ein und demselben Programmcode Daten aus einer Datenbank auslesen und in einer Animation darstellen. Eine genaue Beschreibung zur Installation der Bibliothek und viele Programmcodebeispiele findet man auf der Webseite des Projektes, www.opaque.net/ming/.

Reference

Weitere Informationen zur Ming Library finden Sie unter http://www.opaque.net/ming/. Der Apache Webserver wird unter http://www.apache.org beschrieben, für PHP-Infos besuchen Sie doch mal die Webseite http://www.php.net und für Debian Linux http://www.debian.org.

Einlesen der Uhrzeit und des Datums

Es gibt viele Wege, Datum und Uhrzeit in Flash zu bringen, Flash 5-, Flash 4- und sogar Flash 3-kompatible Wege. Aber es gibt einen Unterschied zwischen der Zeit des Servers und des Users. Wir beschreiben hier vier mögliche Wege.

▸
Saban Ünlü
www.netTrek.de

Wie liest man die aktuelle Uhrzeit und/oder das aktuelle Datum in Flash Version X aus? In diesem Kapitel will ich Ihnen die Möglichkeiten und die eventuellen Probleme, die auftreten können, näher bringen.

Beschäftigt habe ich mich mit diesem Thema, als ich für einen Kunden eine Routine schreiben sollte, die das Last Update immer auf das aktuelle Datum setzen sollte. Das Problem war allerdings, dass ich kein CGI bzw. PHP benutzen durfte, was mich dann auf eine pure JavaScript-Lösung einschränkte.

Mir ist es schließlich gelungen, eine JavaScript-Lösung für alle Systeme zu erstellen. Zwar mit Einschränkungen, aber es funktioniert. Die Einschränkungen beziehen sich auf den Internet Explorer, wenn man am Mac OS-System arbeitet. Hier kann man die SetVariable JavaScript-Funktion nicht einsetzen, und man ist gezwungen, die Variablen bei URL-Aufruf zu übermitteln. Dafür muss aber auch die HTML-Datei dynamisch sein, und es ist folglich keine Synchronisation ohne einen Reload möglich.

Um ein möglichst großes Spektrum der Datums- und Uhrzeitvarianten abdecken zu können, habe ich auch die Variante mit PHP, Perl und Flash 5 erläutert.

Man wird schnell feststellen, dass diese Varianten, insbesondere die Flash 5-Version, wesentlich einfacher, dynamischer und schneller zu handhaben sind.

◀ **Abbildung 1**
Einstellungen für die Zugriffsrechte

Worauf muss man achten?

Bitte beachten Sie, dass die folgenden vier Lösungsmöglichkeiten auf zwei verschiedenen Basen der Uhrzeit- und der Datumsangabe beruhen.

Die JavaScript und die Flash 5-Lösung beziehen sich auf die Daten, die der Client (User-Rechner) zur Verfügung stellt, und die CGI- (Perl) bzw. PHP-Lösungen beziehen sich auf die Daten, die vom Server (Provider-Rechner) bereitgestellt werden. Sollte Ihr Server im Ausland stehen, kann es passieren, dass die gezeigte Uhrzeit nicht mit der Uhrzeit des Users in Deutschland übereinstimmt.

Auch möchte ich darauf hinweisen, dass die laufende Uhr nicht exakt den Sekundentakt einhält. Dies liegt daran, dass jeder Rechner, je nach Prozessorleistung, eine SWF in einer unterschiedlichen Geschwindigkeit abspielt. Dieses fällt bei Standard-SWF nicht unbedingt auf, aber bei rechenintensiven SWFs, wo viele ActionScripts parallel laufen, doch schon eher. Folglich sollte man die hier beschriebenen laufenden Uhrzeiten nicht unbedingt als Stoppuhr benutzen. Eine exakte Zeitdarstellung ist meines Erachtens noch nicht möglich.

Damit die SWF-Uhrzeit nicht allzu stark von der exakten Uhrzeit abweicht, habe ich in regelmäßigen Abständen die Uhrzeit synchronisiert, mehr dazu später.

Daten vom Server

Die Serverdaten habe ich alternativ mit PHP bzw. Perl ausgelesen und in Flash über Load Variable hereingebracht. Diese Lösung ist nur dann möglich, wenn der Provider freie CGI-Scripts bzw. PHP 3 zulässt. Der Vorteil hierbei ist, dass man durch ein erneutes Ausführen der Skripten stets die synchronisierte Uhrzeit hat.

Bitte beachten Sie, dass bei Flash 4 ein Skript nur dann ausgeführt werden kann, wenn die SWF und das auszuführende Skript auf ein und demselben Server liegen.

Auch die Zugriffsrechte müssen so eingestellt sein, dass jeder die Skripten lesen und ausführen darf.

An dieser Stelle möchte ich mich auch noch recht herzlich bei **Stephan Fischer** bedanken, der mir die Grundstruktur des CGI-Scripts zur Verfügung gestellt hat.

Daten vom Client

Die Uhrzeit oder das Datum des Clients in Flash zu bringen, ist etwas komplexer. Die im Folgenden beschriebene JavaScript-Lösung zum Beispiel ermöglicht nur eine Synchronisation der Daten, wenn der User mit Nestcape oder mit Internet Explorer auf einem Windows-System arbeitet. Der Internet Explorer auf den Mac-System unterstützt weder AktiveX noch die SetVariable-Funktion, und folglich ist eine Synchronisation der Daten hier nur durch einen Reload der HTML-Datei zu verwirklichen.

Mit Flash 5 allerdings funktioniert auch die Synchronisation einwandfrei. Ich habe hierfür die ActionScript-Elemente `function`, `getDate` und `array` benutzt und die Rotation der Zeiger über objektbezogene Action-Scripts umgesetzt, mehr dazu später.

Die JavaScript-Variante

Das Skript

Im Folgenden werde ich nun die JavaScript-Programmierungen erläutern, die zur Übergabe der Datums- und Uhrzeitdaten notwendig sind. Anfangen werde ich mit dem Skript, das die Synchronisation erlaubt, und dem wird das Skript für Mac Internet Explorer folgen!

Hier definiere ich die Funktionen `gib_mir()` und `html_geladen()`!

`gib_mir` hat die Aufgabe, die Daten an Flash zu übergeben, detailliert erläutere ich dies später. Damit dies funktioniert, muss dem Flash-Film »gesagt« werden, dass es für JavaScript offen sein muss. Dies macht man durch `swliveconnect="true"`, siehe 071 (für den Internet Explorer) und 075 (für den Navigator). Wichtig ist auch, dass der SWF eine ID beim Internet Explorer (siehe 069) bzw. ein Name bei NC (siehe 075) zugeordnet wird. Dies dient dazu, den Film über JavaScript ansprechen zu können.

`html_geladen` hat lediglich die Aufgabe, die Variable `html` im Flash-Film auf 1 zu setzen.

Dies hat folgenden Hintergrund: Ich möchte gewährleisten, dass der Flash-Film die Funktion **gib_mir()** erst dann verwendet, wenn auch definitiv die gesamte HTML-Datei geladen ist und folglich alle Funktionen definitiv zur Verfügung stehen. Daher lasse ich die SWF so lange loopen, bis die gesamte SWF geladen ist und die Variable **html** auf 1 gesetzt ist (siehe 061 – 063). Damit der Funktionsaufruf **html_geladen** ausgeführt wird, wenn die HTML-Datei geladen ist, setze ich den Aufruf im Body-Tag als **onload** ein (siehe 068)!

```
001 <html><head><meta http-equiv="content-type" content=
"text/html;charset=iso-8859-1">
002 <title>datum_sy</title>
003 <script language="JavaScript1.2"><!--
004  var browser = navigator.appName.indexOf("Microsoft") == -1;
005  var plattform = navigator.platform.indexOf("Win") == -1;
006  if (browser) {var film = "datum";}
007  else {if (plattform) {window.self.location.href = "ohne_syn.html";}}
008
009 var film = "datum";
010 function gib_mir (){
011  currentDate = new Date();
012  year = (currentDate.getYear());
013    if (document.layers){year = year + 1900;}
014
015  month = (currentDate.getMonth() + 1);
016    if (month < 10) {month = "0"+month;}
017      month = month+"."
018      month = month.substring(0,2);
019
020  day = currentDate.getDate();
021     if (day < 10) {day = "0"+day;}
022       day = day+".";
023       day = day.substring(0,2);
024
025  year = year+".";
026  zerlegejahr = year.substring(2,4);
027  year = year.substring(0,4);
028  gesamt_datum = day+"."+month+"."+zerlegejahr;
029  gesamt_datum = gesamt_datum.substring(0,8);
030
```

```
031  houre = currentDate.getHours();
032  minute = currentDate.getMinutes();
033  second = currentDate.getSeconds();
034
035  var fertig = 1;
036    if (browser)
037      {
038   document.embeds[film].SetVariable("year",year);
039   document.embeds[film].SetVariable("mon",month);
040   document.embeds[film].SetVariable("day",day);
041   document.embeds[film].SetVaria-
ble("gesamt_datum",gesamt_datum);
042   document.embeds[film].SetVariable("hours",hour);
043   document.embeds[film].SetVariable("minutes",minute);
044   document.embeds[film].SetVariable("seconds",second);
045   document.embeds[film].SetVariable("status",fertig);
046       }
047   else
048      {   document.all[film].SetVariable("year",year);
049   document.all[film].SetVariable("mon",month);
050   document.all[film].SetVariable("day",day);
051   document.all[film].SetVariable("gesamt_datum",gesamt_datum);
052   document.all[film].SetVariable("hours",houre);
053   document.all[film].SetVariable("minutes",minute);
054   document.all[film].SetVariable("seconds",second);
055   document.all[film].SetVariable("status",fertig);
056       }
057
058 }
059 function html_geladen()
060 {
061   var status = 1
062   if (browser){document.embeds[film].SetVariable("html",status);}
063   else {document.all[film].SetVariable("html",status);}
064  }
065  // -->
066 </script></head>
067
068 <body bgcolor="white" onload="html_geladen()">
069 <object classid="clsid:D27CDB6E-AE6D-11CF-96B8-444553540000"
```

```
    codebase="http://active.macromedia.com/flash2/cabs/swflash
    .cab#version=4,0,0,0" id="datum" width="781" height="213">
070 <param name="movie" value="datum_sy.swf">
071 <param name="swliveconnect" value="true">
072 <param name="quality" value="high">
073 <param name="scale" value="ExactFit">
074 <param name="bgcolor" value="#FFFFFF">
075 <embed src="datum_sy.swf" swliveconnect="true" name="datum"
    quality="high" scale="exactfit" bgcolor="#FFFFFF" WIDTH="781"
    HEIGHT="213" type="application/x-shockwave-flash"
    pluginspage="http://www.macromedia.com/shockwave/download/
    index.cgi?P1_Prod_Version=ShockwaveFlash">
076 </object></body></html>
```

Zeile 001 – 002 sollte allen verständlich sein.

Zeile 003 – 007: Hier kontrolliere ich, ob der User mit dem Mac OS-System den Internet Explorer nutzt und leite ihn, falls ja zu der JavaScript-Variante ohne Synchronisation oder besser gesagt zu der HTML-Datei, die dies beinhaltet, weiter. Ich starte die JavaScript-Umgebung (003) und definiere die Variable `browser` (004) so, dass es den Wert `True` annimmt, wenn der User nicht am Microsoft Internet Explorer arbeitet. Die Variable `plattform` (005) nimmt den Wert `True`, an wenn der User an einem anderen System als dem Windows-System arbeitet.

Mit der IF-Anweisung (006) kontrolliere ich, ob die Variable `browser = True` ist. Falls Nein (dies bedeutet, dass der User nicht mit dem Internet Explorer arbeitet), wird die Variable `film = "datum"` definiert. Die Variable `film` entspricht der `id` bzw. `namen` (siehe 069 und 075) der SWF.

Falls ja, schaue ich, ob der Internet Explorer auch auf dem System läuft, das die Synchronisation über JavaScript unterstützt . Die IF-Anweisung (007) prüft also, ob der User mit einem Windows-System arbeitet, und leitet ihn mit `window.self.location.href =` weiter auf die alternative HTML-Seite (ohne_syn.html), falls dies nicht zutrifft.

Zeile 009: Nun, da ich gewährleisten kann, dass die Synchronisation funktionieren wird, definiere ich die Variable `film = "datum"` als globale Variable und kann folglich allen Funktionen in der HTML-Datei die `id` bzw. `namen` (siehe 069 und 075) der SWF mitteilen. Zur Funktion `gib_mir()` siehe 010 bis 058.

In Zeile 011 definiere ich das Objekt `currentDate` über `new Date()`. Dies kann man sich am besten vorstellen als Definition eines Arrays, aus dem man über `getYear(), getDate(), getMonth()...`etc. Jahr, Tag, Mo-

nat etc. als Teilinformation entnimmt. Nähere Informationen bei SelfHTML (http://www.teamone.de/selfaktuell/).

Zeile 012: Über `getYear` hole ich aus dem Objekt `currentDate` die Jahreszeit und übergebe sie der Variablen `Year`. Das Ergebnis wird immer in Jahrzehnten berechnet, d.h. `getYear = RealeJahresZahl-1900`, z.B. bei 1999 wäre das Ergebnis 99 und bei 2000 folglich 100.

Zeile 013: Die IF-Anweisung sorgt mit +1900 folglich für die richtige vierstellige Jahreszahl.

Zeile 015: Über `getMonth` hole ich aus dem Objekt `currentDate` die Monatszahl und übergebe sie der Variablen `month`. Das `+1` folgt, weil die Ausgabe für `getMonth` wie folgt definiert ist: 0 entspricht Januar ... 11 entspricht Dezember. Da ich nur die Zahlen brauche und Januar bekanntlich 1 und nicht 0 ist, folgt hinter der `getMonth`-Anweisung das `+1.`

Zeile 016: Die IF-Anweisung hat hier eine rein kosmetische Funktion. Sie checkt, ob die Monatszahl einstellig ist und setzt eine 0 vor die 1, falls `month` kleiner ist als 10.

Zeile 018: Der folgende Substring stellt lediglich sicher, dass `month` wirklich zweistellig übergeben wird. Die Substring-Anweisung funktioniert wie folgt: `Variablenname.substring(startIndex, stopIndex);`

Sie gibt einen Teilbereich eines Strings (Zeichenkette) zurück. Um den Teilbereich zu definieren, braucht man den `startIndex`, der ein Integer (ganze Zahl) ist, der die erste Stelle angibt, und einen `stopIndex`, der das Ende des Teilbereiches definiert. Will man einen String von dem ersten Zeichen aus beschneiden, ist der `startIndex = 0.` Der `startIndex` ist so zu setzen, dass das erste zu integrierende Zeichen mit dessen Position –1 ist. Am einfachsten ist es mit einem Beispiel zu erläutern: Wenn Variablenname z.B. = 2000, `startIndex = 2` und `stopIndex = 4` ist, dann ist das Ergebnis des Substrings 00.

Zeile 017: Die davor gesetzte Zeile `month = month+"."` dient lediglich dazu, dass der Substring-Befehl auch definitiv funktionieren kann. Da `month` ja ein Integerwert ist, muss ich über +».« einen String daraus machen, damit Substring auch angewandt werden kann.

Zeile 020: Über `getDay` aus dem Objekt `currentDate` die Tageszahl entnehmen und an die Variablen `day` übergeben.

Zeile 021: Die IF-Anweisung und die folgenden Zeilen (022 – 023) erfüllen den gleichen Zweck wie die Anweisung im Abschnitt Monat.

Zeile 025: siehe 017

Zeile 026 und 027: Hier erzeuge ich die zwei- und vierstellige Jahreszahl mit Hilfe von Substring `Zerlegejahr` = zweistellig und `year` = vierstellige Jahreszahl.

Zeile 028 und 029: Mit diesen beiden Zeilen habe ich das gesamte Datum im Format dd.mm.yy zusammengestellt und in die Variable `gesamt_datum` gesetzt. Dies mache ich, indem ich die Tageszahl mit dem String `"."`, der Monatszahl, mit dem String `"."` und der zweistelligen Jahreszahl durch + (Addition einer Zeichenkette) verbinde. Der Substring übernimmt lediglich die Kontrollfunktion, dass das Format auch wirklich nur achtstellig ist.

Zeile 031: Über `getHours` entnehme ich aus dem Objekt `currentDate` die Stundenzahl und übergebe sie der Variablen `hour`. Auf die kosmetische IF-Anweisung habe ich hier (ebenso bei `minute` und `second`) verzichtet, weil ich die Werte in Flash für die laufende Uhrzeit als Integer benötige. Durch +»0« würde ich den Integer ja in einen String umwandeln und könnte folglich nicht mehr mathematisch mit dem Wert weiterarbeiten. Die Kosmetik (»0« bei Werten < 10) habe ich in Flash umgesetzt.

Zeile 032 und 033 funktionieren mit demselben Prinzip wie `getHours`!

Der Wert der Minute wird über `getMinute()` an die Variable `minute` und der Wert der Sekunde über `getSeconds` an die Variable `second` übergeben.

Zeile 035: Hier definiere ich die Variable `fertig = 1`. Dies ist lediglich eine Kontrollvariable; ich übergebe den Wert von `fertig` an Flash als Variablenwert von `status` (siehe 045 und 055). Dies mache ich, um gewährleisten zu können, dass alle Daten an Flash übermittelt wurden. Dafür lasse ich die SWF so lange in einer Schleife laufen, bis status = 1 ist.

Zeile 036–055: Hier werden nun die zuvor errechneten Werte an Flash übermittelt. `SetVariable()` ist die JavaScript-Funktion, die diese Aufgabe für uns erfüllen soll. Die Syntax der Funktion ist denkbar einfach und ähnelt sehr der Flash-Syntax: `SetVariable(Variablenname,VariablenWert)`

Wenn wir beispielsweise `SetVariable("saban","Ünlü")` nehmen, ist das äquivalent mit einem `Set Variable: "Saban" = "Ünlü"`, das ich in einer FLA auf der Grundebene (_levelo) definiere.

Ich kann auch Variablen an MovieClip-Instanzen übergeben, indem ich den Pfad vor den Variablenname setze, genau so, wie es uns aus Flash bekannt ist.

Somit ist es z. B. möglich, in der Instanz `"Movie"` die Variable `"Test"` zu setzen, indem man den Variablenname als `"/Movie:Test"` setzt.

Wir müssen nun nur noch ein Ziel ansetzen, auf das `SetVariable` angesetzt werden soll. Um dies zu ermöglichen, haben wir, wie zuvor erläutert, die ID bzw. den Namen der SWF definiert. Um nun unter Internet Explorer der mit dem `<object>`-Tag eingebundenen SWF eine Variable zuordnen zu können, müssen wir folgende Syntax verwenden:

`document.all[ID].SetVariable(Variabe+lenname,VariablenWert);`

Dadurch wird in dem gesamten Dokument nach der SWF mit der passenden ID gesucht und auf dies die Funktion `SetVariable` ausgeführt.

Analog wird es mit dem Navigator über die mit dem `<embed>`-Tag eingebundene SWF und deren hier als Parameter `NAME` definierte Bezeichnung ausgeführt. Die Syntax hier ändert sich nur in dem Teil vor `SetVariable()`:

```
document.embeds[NAME].SetVariable(Variabelenname,VariablenWert);
```

Folglich hat die in Zeile 036 angebrachte IF-Anweisung die Aufgabe zu unterscheiden, ob es sich bei dem User-Browser um den Internet Explorer oder den Navigator handelt. Dementsprechend werden beim Internet Explorer über

```
document.all[ID].SetVariable(Variablenname,VariablenWert);
```

(siehe Zeile 048–055) oder beim NC über

```
document.embeds[NAME].SetVariable(Variablenname,VariablenWert);
```

(siehe 038–045) die zuvor errechneten Daten übergeben.

Wichtig: Ich möchte hier noch einmal, wie bereits oben erläutert, daran erinnern, dass der Parameter `swliveconnect="true"` gesetzt sein muss, damit die SWF die JavaScript-Kommandos akzeptiert.

Zeile 059–064: Die Funktion `html_geladen()` und deren Aufgabe hatte ich bereits oben erläutert. Wie sie ihre Aufgabe erfüllt, dürfte jetzt auch durch die Erläuterung von `SetVariable` deutlicher geworden sein.

Ich definiere in 061 die Variable `status = 1`, um diesen Wert dann in 062–063 als Wert der Variablen `html` in Flash zu setzen.

Zeile 064–076: Der Rest der HTML-Datei müsste eigentlich jedem bekannt sein. Nachdem der `</Script>`-Tag beendet ist, folgt der `<Object>`- und `<embled>`-Tag äquivalent, wie es Flash bei der Veröffentlichung einer FLA mit einer HTML-Datei macht.

Probleme

Jetzt kommen wir zu dem Part, in dem das zuvor Beschriebene in der Praxis nicht funktioniert (um genau zu sein, bei Internet Explorer auf Mac

OS). Diese Möglichkeit haben wir ja bereits oben berücksichtigt und eine Weiterleitung, für diesen Fall, auf die HTML-Datei namens »ohne_syn.html« eingerichtet.

Was nun in dieser Datei anders ist als in der synchronisationsfähigen index.html, wird im Folgenden erläutert.

»ohne_syn.html« ist selbstverständlich nicht nur bei Mac Internet Explorer, sondern auch auf allen anderen Standardbrowsern und -systemen lauffähig.

Nun, da ja, wie schon oben beschrieben, die `SetVariable` nicht auf allen Systemen funktioniert, habe ich es so gelöst, dass ich der SWF beim Aufruf die Variablen übergebe, z. B. http://www.uenlue.de/datum/datum.swf?Variablenname=Wert

Den Ausdruck `Variablenname=Wert` können Sie einfach so interpretieren, dass Sie im ersten Frame ein `SetVariable` (`"Variablenname"=Wert`) gesetzt haben.

Folglich bedeutet `?v1=wert1&v2=wert2` nichts anderes, als dass auf _levelo Frame1 die Variablen `v1` und `v2` wie folgt gesetzt werden:

```
Set Variabel ("v1" = wert1);
Set Variabel ("v2" = wert2);
```

Da ja die Zeit bekanntlich ein dynamischer Wert ist, muss folglich auch die HTML-Datei welche die SWF eingebettet hat, dynamisch sein. D. h. ich ermittele die Daten für die Uhrzeit und das Datum, und über document.writeln. schreibe ich dann dynamisch die HTML-Tags, welche die SWF mit der Variablen `Übergabe` eingebunden hat.

Explizit sieht das wie folgt aus:

```
<HTML><HEAD>
<TITLE>Datum und Uhrzeit --- von saban@uenlue.de</TITLE>
```

Wichtig: Das Skript muss im Head stehen, damit es sofort bei Start (Laden) der Seite ausgeführt werden kann.

```
<script language="JavaScript">
<!--
currentDate = new Date();
```

Siehe JavaScript-Erläuterung für Zeile 011.

```
year = (currentDate.getYear());
  if (document.layers){
  year = year + 1900;   }
```

Siehe JavaScript-Erläuterung für Zeile 012–013.

```
month = (currentDate.getMonth() + 1);
  if (month < 10) {month = "0"+month;}
  month = month+"."
  month = month.substring(0,2);
```

Siehe JavaScript-Erläuterung für Zeile 015–018.

```
day = currentDate.getDate();
   if (day < 10) {day = "0"+day;}
   day = day+".";
   day = day.substring(0,2);
```

Siehe JavaScript-Erläuterung für Zeile 020–023.

```
year = year+".";
zerlegejahr = year.substring(2,4);
year = year.substring(0,4);
```

Siehe JavaScript-Erläuterung für Zeile 025–027.

```
gesamt_datum = day+"."+month+"."+zerlegejahr;
gesamt_datum = gesamt_datum.substring(0,8);
```

Siehe JavaScript-Erläuterung für Zeile 028–029.

```
hour = currentDate.getHours();
minute = currentDate.getMinutes();
second = currentDate.getSeconds();
```

Siehe JavaScript-Erläuterung für Zeile 031–033.

Lassen Sie uns kurz memorieren, wie ich nun die Variablen bei SWF-Aufruf übergeben wollte:

```
SWFname.swf?Variable1=Wert1&Variable2=Wert2
```

Diese URL werde ich jetzt im Folgenden durch Teilabschnitte zusammensetzen.

Dabei sind die Komponenten url1...url7 die, welche die Variablennamen beinhalten. Einzige Besonderheit ist url1, die auch den SWF-Namen innehat.

```
   url1="'datum.swf?year=";
   url2="&mon=";
   url3="&day=";
url4="&gesamt_datum=";
   url5="&hours=";
   url6="&minutes=";
   url7="&seconds=";
url_gesamt=url1+year+url2+month+url3+day+url4+gesamt_datum+url5
+hour+url6+minute+url7+second;
```

`url_gesamt` entspricht dann folgendem Ausdruck:

```
url_gesamt='datum.swf?year=year&mon=month&day=day&gesamt_datum
=gesamt_datum&hours=hour&minutes=minute&seconds=second;
```

Die Werte entsprechen den oben errechneten Variablenwerten für die Datums- und Uhrzeitkomponenten.

Verständlicher ist der Ausdruck, wenn man die HTML-Datei in Netscape öffnet. Wenn `gesamt_url` dann wie folgt lautet:

```
datum.swf?year=2000&mon=09&day=26&gesamt_datum=26.09.00&hours=14&
minutes=28&seconds=50
```

bedeutet dies, dass auf _levelo (der Grundebene) von datum.swf die Variablen (äquivalent wie durch `Set Variable`) `year = 2000 , mon = 09 , day = 26 etc.` gesetzt wurden.

Nun, wie schafft man es jetzt, diese Werte über die URL an Flash weiterzugeben? Die Problematik besteht ja darin, dass die Werte dynamisch sind und eine HTML-Datei »statisch«!

Die Lösung ist, ich schreibe die HTML-Syntax erst, wenn ich alle Werte errechnet habe, die ich für die URL-Übergabe benötige.

Ich kann eine HTML-Datei dynamisch über JavaScript erstellen, indem ich die `Write()`-Funktion benutze. Mit Write kann ich Zeichenketten, Variablen, JavaScript-Ausdrücke etc. auf der aktuellen HTML-Seite ausgeben lassen. Hierfür muss ich nur ein Ziel für die Write-Funktion definieren.

Das heisst, `document.write("hallo");` bedeutet, dass im aktuellen HTML-Dokument (Ziel:document.) ein »hallo« auf dem Bildschirm erscheinen wird.

Ein anderes Beispiel wäre `parent.frames[FRAMENAME].write ("hallo");` – hier würde das Hallo im Nachbarframe namens `FRAMENAME` erscheinen.

Eine Kombination aus Zeichenkette und beispielsweise Variablenwerten will ich mit Hilfe der `Writeln();`-Funktion erläutern. Der Unterschied zwischen den Funktionen `write` und `writeln` liegt lediglich darin, dass bei `writeln` nach dem geschriebenen Ausdruck ein Zeilenumbruch folgt, z. B.

```
name = "Saban";
dm = 15;
document.writeln("Das Mitglied mit dem Namen: " + name)
document.writeln("hat bereits " + dm + " DM bezahlt.")
```

Auf dem Bildschirm erscheint dann folglich:

Das Mitglied mit dem Namen: Saban hat bereits 15 DM bezahlt.

Wie löse ich es jetzt, wenn ich Anführungsstriche im Ausdruck von der `write()`-Funktion haben will? Diese Frage lässt sich am einfachsten durch ein zweites Beispiel erläutern:

```
document.write("<a href=\"datei.htm\">Das ist ein Link</a>");
```

oder

```
document.write("<a href='datei.htm'>Das ist ein Link</a>");
```

Beide Ausdrücke würden folgendem HTML-Tag entsprechen

```
<a href="datei.htm">Das ist ein Link</a>
```

und wie man sieht, habe ich im `write`-Ausdruck Anführungsstriche, ohne sie tatsächlich genutzt zu haben.

Würde ich den Tag so schreiben, wie wir es gewohnt sind, also

```
document.write("<a href="datei.htm">Das ist ein Link</a>");
```

so würde der Ausdruck von `write` wie folgt aussehen: `<a href=`

Im Klartext: Möchte ich Anführungsstriche als Ausdruck der `write`-Funktion haben, so muss ich diese im `write`-Parameter als `\"` bzw. `'` schreiben.

Wenn nun vor der `document.write()`-Anweisung ein `document.open("text/html")` steht, passiert Folgendes: Es wird ein Dokument zum Schreiben geöffnet, das Dokument befindet sich immer noch in der aktuellen HTML-Datei und bewirkt somit, dass der bisherige Inhalt geleert wird. Der Parameter `"text/html"` hat zur Folge, dass das »neue« Dokument zum Schreiben von HTML geöffnet wird.

Analog dazu definiert die Funktion `close()` das Ende des mit `open()` geöffneten Dokuments.

> **Weiterführende Informationen**
> Detaillierte Informationen zu den eben beschriebenen Funktionen finden Sie bei SelfHTML (http://www.teamone.de/selfaktuell/).

Mit Hilfe der oben beschriebenen Funktionen werde ich also eine dynamische HTML-Datei erzeugen. Dynamisch sind hier eigentlich nur die Variablen und deren Werte, die ich beim URL-Aufruf definieren möchte. Verständlich ist, dass hierdurch keine Synchronisation der Uhrzeit möglich ist, da man hierfür ein Reload ausführen müsste, um die Variablen neu übergeben zu können, was ja nicht sinnvoll wäre.

Die dynamische HTML-Datei, welche die SWF eingebunden hat, ist bis auf den Sourcecode der SWF, identisch mit der, die Flash beim Veröffentlichen erzeugen würde. Der Sourcecode ist unterschiedlich, weil er die Variablen und deren Werte beinhaltet.

Die »dynamische« HTML-Datei

```
document.open("text/htm")

document.writeln("<html><head><title>DATUM und Uhrzeit --- von
saban@uenlue.de </title></head>")

document.writeln("<body bgcolor='#FFFFFF'>")

document.writeln("<OBJECT classid='clsid:D27CDB6E-AE6D-11cf-96B8-
444553540000' codebase='http://download.macromedia.com/pub/
shockwave/cabs/flash/swflash.cab#version=4,0,2,0' ID=datum
WIDTH=781 HEIGHT=213>")
```

```
document.writeln("<PARAM NAME= movie VALUE="+url_gesamt+"'>")

document.writeln("<PARAM NAME=name VALUE=datum><PARAM NAME=quality
VALUE=high><PARAM NAME=scale VALUE=exactfit><PARAM NAME=bgcolor
VALUE=#FFFFFF><PARAM NAME=swliveconnect VALUE=True>")

document.writeln("<EMBED src="+url_gesamt+"' quality=high
swliveconnect=true name=datum scale=exactfit bgcolor=#FFFFFF
WIDTH=781 HEIGHT=213 TYPE='application/x-shockwave-flash'
PLUGINSPAGE='http://www.macromedia.com/shockwave/download/
index.cgi?P1_Prod_Version=ShockwaveFlash'></EMBED>")

document.writeln("</OBJECT>")

document.close()
//-->
</script>
</HEAD>
<BODY bgcolor="#FFFFFF"></BODY>
</HTML>
```

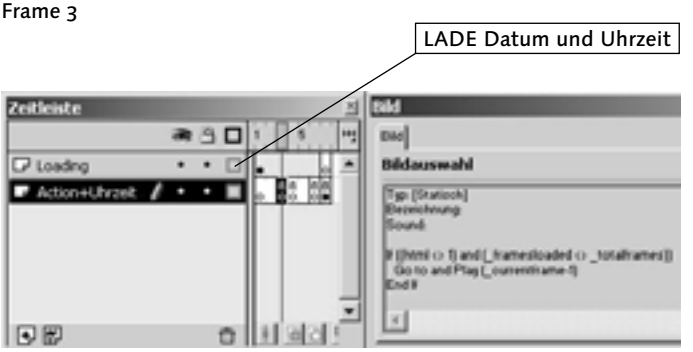
Wenn ich jetzt ohne_syn.html direkt in Netscape öffne, wird dieser mir bei [Ansicht]-[Quellcode] folgenden HTML-Code zeigen:

```
<HTML>
<HEAD>
<TITLE>Datum und Uhrzeit --- von saban@uenlue.de</TITLE>
<HTML><HEAD><TITLE>DATUM TEST</TITLE></HEAD>
<BODY bgcolor='#FFFFFF'>
<OBJECT classid='clsid:D27CDB6E-AE6D-11cf-96B8-444553540000'
codebase='http://download.macromedia.com/pub/shockwave/cabs/flash
/swflash.cab#version=4,0,2,0' ID=datum WIDTH=781 HEIGHT=213>

<PARAM NAME= movie VALUE='datum.swf?year=2000&mon=09&day=26&
gesamt_datum=26.09.00&hours=14&minutes=28&seconds=50'>

<PARAM NAME=name VALUE=datum><PARAM NAME=quality VALUE=high>
<PARAM NAME=scale VALUE=exactfit><PARAM NAME=bgcolor
VALUE=#FFFFFF><PARAM NAME=swliveconnect VALUE=True>
```

Grundebene von datum_sy.fla
Frame 3



◀ Abbildung 2
Die Grundebene von datum_sy

```
<EMBED src=' datum.swf?year=2000&mon=09&day=26&gesamt_datum=
26.09.00&hours=14&minutes=28&seconds=50' quality=high
swliveconnect=true name=datum scale=exactfit bgcolor=#FFFFFF
WIDTH=781 HEIGHT=213 TYPE='application/x-shockwave-flash'
PLUGINSPAGE='http://www.macromedia.com/shockwave/download/
index.cgi?P1_Prod_Version=ShockwaveFlash'></EMBED>

</OBJECT>
</HEAD>
<BODY bgcolor="#FFFFFF">
</BODY>
</HTML>
```

Wie wir sehen, werden die Variablen mit dem Sourcecode (URL-Aufruf) der SWF übergeben.

Die Flash-Programmierung

Nun kommen wir zu den FLAs, welche die Variablen aufnehmen und weiterverarbeiten sollen. Beginnen will ich mit der FLA, die synchronisierbar ist, also datum_sy.fla in die index.html eingebunden hat.

Die Grundebene von datum_sy enthält zwei Layer, »Loading« und »Action+Uhrzeit«. In »Loading« befindet sich lediglich das Textfeld mit dem Inhalt »Lade Datum und Uhrzeit«, das wir in Abbildung 3 sehen.

Abbildung 3 ▶
Der MovieClip time in Frame 7

Grundebene von datum_sy.fla
Frame 7

In Frame 3 kontrolliere ich mit der IF-Anweisung, ob die Variable `html` ungleich eins ist und ob `_framesloaded` ungleich `_totalframe`.

Falls dieses der Fall ist, also der Film noch nicht geladen und/oder html noch nicht durch die JavaScript-Funktion `html_geladen` auf 1 gesetzt ist, wird durch `Go to and Play` (`_currentframe-1`) der Film zurück zu Frame 2 gesetzt und abgespielt.

Sollte nun der Film geladen sein, also `_framesloaded = _totalframe`, und die Statusvariable html = 1 sein, wird die IF-Anweisung ignoriert und der Film geht weiter zum nächsten Frame.

In Frame 4 lassen wir durch die in HTML eingebettete JavaScript-Funktion `gib_mir()` die Variablen an Flash übermitteln. Um eine JavaScript-Funktion aus Flash aufrufen zu können, müssen wir einfach: `GetURL ("javascript:gib_mir()"`) ausführen. Der Ausdruck `javascript:` hat hier die gleiche Wirkung, die er in der HTML-Umgebung beim `href`-Tag hat. D. h. er verdeutlicht dem Browser, dass nun trotz des Linkaufrufs die Ausführung einer JavaScript-Anweisung gemeint ist.

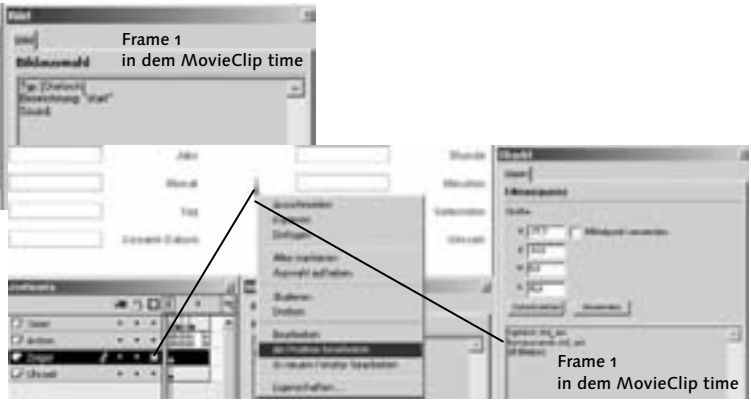
Wir wollen uns noch einmal kurz an `gib_mir()` erinnern. Hier haben wir nicht nur die Variablen für das Datum und die Uhrzeit gesetzt, sondern auch die Variable `status = 1`.

Mit der IF-Anweisung an dieser Stelle lasse ich den Film so lange in den vorherigen Frame springen und abspielen, bis `status` nicht mehr ungleich 1 ist.

Das ist dasselbe Prinzip wie bei der Variablen `html`, nur dass die IF-Anweisung nicht mit einer weiteren UND-Bedingung verknüpft ist.

Ich halte also den Flash-Film so lange in der Schleife gefangen, bis alle Variablen der SWF übergeben wurden und somit `status= 1` ist.

▲ Abbildung 4
Die Layer des MovieClip time



▲ Abbildung 5
Layer »Zeiger« von MovieClip time

In Frame 7 sind wir dann am eigentlichen Ziel angelangt. Hier finden wir alle Textfelder, die später mit den Variablen gefüllt werden. Des Weiteren finden wir hier den MovieClip `time` (Abbildung 3 im markierten Viereck); dieser übernimmt die Funktion der laufenden Uhrzeit.

Der MovieClip `time` hat, wie man in Abbildung 4 sieht, vier Layer. Im ersten Layer »Timer« befindet sich im ersten Frame ein Label `= "start",` das für ein späteres `Go to (Label)` notwendig ist.

Im Layer »Zeiger« von MovieClip `time` findet man die MovieClips `std_ani, min_ani und sek_ani`. Das sind einfach die Zeiger der **Analoguhr**, deren Funktion ich im Folgenden erläutere. Wichtig hier zu erwähnen ist noch, dass ich den MovieClip-Instanznamen gegeben habe. Dieses macht man ja, indem ein Doppelklick auf den MovieClip ausgeführt wird und man in das Feld »Instanz« des geöffneten Fensters den Namen einträgt. Um die Sache zu erleichtern, sind die Instanznamen gleich den MovieClip-Namen von `std_ani, min_ani und sek_ani.`

In MovieClip `std_ani` habe ich ein Symbol eingefügt, das eine senkrechte Linie beinhaltet und dessen Mittelpunkt ich zentriert unten angelegt habe (siehe Abbildung 5).

Wenn ich nun auf dieses Symbol ein Bewegungstweening im Uhrzeigersinn anwende, wird das untere Ende (Mittelpunkt) immer an derselben Stelle stehen bleiben und der Rest der Linie rechts um den Mittelpunkt herum rotieren.

Somit habe ich eine Animation für einen Uhrzeiger erstellt. Die Tweening-Länge habe ich nun wie folgt definiert: In Frame 1 hat der Zei-

Abbildung 6 ▸
Variablen werden in interne Variablen gesetzt

ger seine Ursprungsposition, und dies entspricht der Stunde 0. Da bei 12:00 Uhr der Zeiger wieder dieselbe Position haben muss, setze ich den Zeiger nach zwölf Frames, also in Frame 13, noch ein Mal. Wieder nach zwölf Frames, in Frame 25, hätten wir dann wieder 00:00 Uhr. Wenn wir jetzt in Frame 1 und 13 ein Bewegungstweening im Uhrzeigersinn mit einer Umdrehung einstellen und uns die Animation ansehen, sehen wir, dass der Zeiger 2 Mal rotiert und so, bezogen auf die Stunden, einen ganzen Tag simuliert.

Es ist selbstverständlich auch machbar, dass man nur ein Tweening von Frame 1 bis Frame 25 verwendet und hier lediglich darauf achtet, dass die Drehung im Uhrzeigersinn auf 2 gestellt ist.

Analog dazu habe ich `sek_ani` und `min_ani` erstellt. Der Unterschied hier ist lediglich, dass die Periode einer Sekunde und einer Minute nach 60 Frames vorüber ist. D.h. das Tweening mit der einmaligen Drehung im Uhrzeigersinn liegt hier in Frame 1 und endet in Frame 61.

Deutlich wird das, wenn man sich einfach die MovieClips `sek_ani` und `min_ ani` in der beiliegenden FLA betrachtet.

Warum nun dieses Tweening? Nun, je nach Stunde, Minute bzw. Sekunde springe ich später durch `Tell Target` und `Go To (Frame)` an die entsprechende Position des MovieClips und habe somit automatisch die Positionen der Zeiger gesetzt.

In MovieClip `time` sehen wir nun in Frame 1 des Layers »Action«, wie wir mit den Variablen, die wir über die JavaScript -Funktion `gib_mir()` erhalten haben, fortfahren.

Wir fangen damit an, dass wir die erhaltenen Variablen `hours`, `minutes` und `seconds` in interne Variablen `st_i` (für die Stunde), `m_i` (für die Mi-

nute) und s_i (für die Sekunde) setzen. Dies geschieht einfach, wie in Abbildung 6 zu ersehen, durch Set Variable: "st_i" = ../hours.

In diesem Falle wird der Wert von hours in die Variable st_i gesetzt. Warum ich dabei ../:hours verwende, hat folgenden Grund: Die Action, die ich hier ausführe, liegt in MovieClip time, die Variable hours allerdings in der Grundebene (_level0). Mit ../ gehe ich quasi aus dem MovieClip time heraus (und zwar exakt eine Ebene tiefer, also in die Grundebene). Durch das ":", das dem ../ folgt, »weiß« Flash nun, dass ich in dieser Ebene lediglich eine Variable bzw. deren Wert aufgreifen will. Die Syntax _level0:hours würde hier beispielsweise das Gleiche bewirken. Die Syntax mit dem / funktioniert ähnlich wie bei HTML (oder Dos \ ), um auf tiefer bzw. höher liegende Dateiordner zuzugreifen. Nur dass bei Flash nicht Ordner, sondern Instanzen bzw. Ebenen angesteuert werden.

Äquivalent zu st_i setze ich die Variablen m_i und s_i.

Im Folgenden werde ich durch Tell Target und Go To den Minutenzeiger positionieren. Wir wollen uns noch mal kurz erinnern. Der MovieClip std_ani, der im MovieClip time liegt, hat ein Symbol, das von Frame 1 bis Frame 13 und in Frame 13 bis Frame 25, durch ein Bewegungstweening im Uhrzeigersinn, zwei Mal um seine eigene Achse rotiert. Dadurch steht der Stundenzeiger bei Frame 1 auf Stunde 0, bei Frame 13 auf Stunde 12 und bei Frame 25 wieder auf Stunde 0. Wir sehen hier einen mathematischen Zusammenhang, nämlich dass die Stundenposition des Zeigers der Framezahl + 1 entspricht. Daraus folgt, dass – wenn der Stundenzeiger auf Stunde 15 stehen soll – in MovieClip std_ani der Frame auf der Framezahl 15 + 1, also auf Frame 16, stehen muss.

Folglich gilt: Frame = Stunde + 1 (z. B. Stunde = 13, d. h. Frameposition bei 14)

Nun müssen wir das nur noch in ActionScript umsetzen. Dies machen wir durch folgende Syntax:

Durch den Tell Target gehen wir in den MovieClip std_ani.

Mit Go to and Stop((../:st_i)+1) setzen wir jetzt die Stundenzeigerposition.

Der Frame [(../:st_i)+1], zu dem wir springen, ist durch den oben beschriebenen mathematischen Zusammenhang (Frame = Stunde + 1) wie folgt zu setzen:

```
Begin Tell Target ("std_ani")
   Go to and Stop ((../:st_i)+1)
End Tell Target
```

Ich nehme den aktuellen Wert der Stunde aus der Variable `st_i`, die in MovieClip `time` definiert wurde. Dadurch, dass wir uns jetzt mit Tell Target in `std_ani` befinden, müssen wir wieder eine Ebene tiefer, folglich auch hier der Ausdruck `../:`

Den Wert erhöhen wir dann um 1 und haben dann die Framezahl, zu der wir springen müssen, damit der Stundenzeiger auch wirklich an der richtigen Position steht. Mit demselben Prinzip setzen wir auch die Zeigerposition für die Minute. Wenn man sich `min_i` und `sek_i` anschaut, sieht man, dass diese ebenfalls demselben mathematischen Zusammenhang folgen wie `std_ani.`

Die Positionierung der Zeiger ist aber nicht das Einzige, was in MovieClip `time`, in Frame 1 des Layers »Action« gemacht wird.

Wie wir sehen, machen wir hier auch wie zuvor schon angesprochen die »kosmetische« Überarbeitung der Zahlenwerte für Stunde, Minute und Sekunde. Da wir wollen, dass die Stunde bei Werten < 10 im Format 00...09 ausgegeben werden soll und dies auch für Minute und Sekunde gilt, folgen die »kosmetischen« IF-Anweisungen.

Bevor ich diese IF-Anweisungen detaillierter erläutere, möchte ich nur kurz eine Variante der Variablenwertdefinition ansprechen. Ich kann den Wert einer Variablen als Ausdruck angeben, um nummerische Werte, andere Variablenwerte oder Kombinationen aus Zeichen und Werten zuordnen zu können. Ich möchte dies an Beispielen zeigen und auch gleich fortfahren:

```
Set Variable: "v1" = 1
Set Variable: "v2" = "Der Wert von "&"v1 entspricht - "
Set Variable: "v3" = v2&v1&" - den nummerische Wert eins"
```

Hier sehen wir:

- Die nummerische Definition der Variablen v1, die den Wert 1 erhält.
- Die Definition der Zeichenfolge v2, die aus der Kombination der Zeichenfolgen »Der Wert von« und »v1 entspricht –«. Im Gesamten entspricht dies : »Der Wert von v1 entspricht –«.
- Die Variable v3 ist Zeichenfolge aus der Kombination von v2, v1, und der Zeichenfolge »– den nummerischen Wert eins«. Im Gesamten entspricht der Wert von v3: »Der Wert von v1 entspricht – 1 – dem nummerischen Wert eins.«

Wenn die Variable `m_i` kleiner als 10 ist und (was ja bei einem Integer eigentlich zwingend) die Länge der Variable `m_i` kleiner 2 ist, dann soll der Variablen `m` der Wert 0 und der Wert m_i, übergeben werden. Wichtig

◄ **Abbildung 7**
Im Ausdruck müssen Anführungsstriche manuell eingegeben werden

hier zu erwähnen ist, dass `m` hier kein Integerwert mehr ist, sondern durch `"0"&` (da 0 in Anführungsstrichen nicht eine Zahl, sondern ein Zeichen ist) zu einem String wird.

Beispiel m_i = 9, d.h. m = »09« – ein String

Ansonsten wird dem Wert von `m` der Wert von `m_i` übergeben.

```
If ((m_i < 10) and (Length (m_i) < 2))
   Set Variable: "m" = "0"&m_i
Else
   Set Variable: "m" = m_i
End If
If ((st_i < 10) and (Length (st_i) < 2))
   Set Variable: "st" = "0"&st_i
Else
   Set Variable: "st" = st_i
End If
```

Äquivalent dazu wird auch die Variable `st` gesetzt. Warum wir hier mit zwei Werten für die Stunde, Minute und Sekunde arbeiten, die eigentlich ein und dasselbe aussagen, ist verständlich, wenn man bedenkt, dass bei Werten <10 die Variable zu einem String umgewandelt wird. Mit Strings kann man aber nicht mehr mathematisch weiterarbeiten, da diese Zeichen sind und keine Zahlen, und folglich sind diese Werte für den Verlauf der FLA nur noch visuell zu gebrauchen. Visualisiert werden die Werte, wie wir im unteren Bild sehen können, in der laufenden Uhrzeit (Layer:Uhrzeit).

Abbildung 8 ▸
Frame 2 des MovieClip time

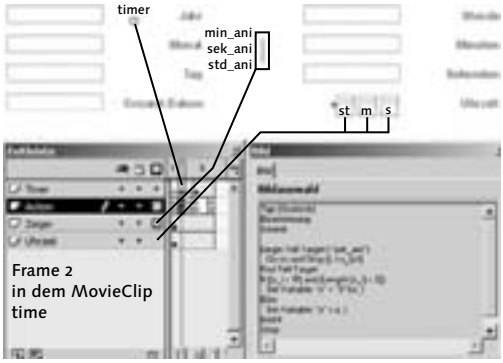


Abbildung 9 ▸
Die Funktion des MovieClip »timer«

In Frame 2 des MovieClip **time** sehen wir nun, dass die Sekundenzeigerposition und die »Kosmetik« von s bearbeitet wird. Das funktioniert mit demselben Verfahren, das ich oben bei **std** und der Stundenzeigerposition erläutert habe. Anschließend folgt ein **STOP**.

Nun werden sich viele fragen, warum hier auf einmal der Film gestoppt wird. Die Frage beantwortet sich quasi von selbst, wenn wir uns Abbildung 9 betrachten, denn hier wird die Funktion des MovieClip **timer** erläutert. Timer können wir bereits in Abbildung 10 sehen, es ist der MovieClip, der im Layer »Timer« auf Frame 2 liegt, zu erkennen an der Linie, die von diesem Frame zum MovieClip **timer** führt.

In Frame 1 von MovieClip **timer**, der in MovieClip **time** auf dem Layer »Timer« an Frameposition 2 liegt, sehen wir, dass der Variablen **StartTimer** der Wert von **GetTimer** zugeordnet wird.

Klartext: Hier wird der Variablen **StartTimer** der Wert zugeordnet, welcher der Laufzeit des SWF entspricht! **GetTimer** wird immer in Millisekunden ausgegeben. Wenn also der Wert von **StartTimer** = 1000 ist, bedeutet dies, dass das SWF bereits seit einer Sekunde läuft.

Im Frame 2 von MovieClip **timer** folgt nun folgende IF-Anweisung, die prüfen soll, ob eine Sekunde abgelaufen ist.

```
If ((GetTimer-StartTimer) >= 999)
   Begin Tell Target ("../")
```

◀ **Abbildung 10**
Frame 2 von MovieClip timer

```
      Play
   End Tell Target
End If
Go to and Play (2)
```

Wenn die Laufzeit – StartTimer (StartTimer, die in Frame 1 aufgenommen wurde) größer oder gleich 999 Millisekunden, dann setze Ziel eine Ebene tiefer (mit `Tell Target`) und spiele diese Ebene ab. Nach der IF-Anweisung gehe zu Frame 2 und spiele ab.

Hier geschieht Folgendes: Nachdem ich in Frame 1 die Laufzeit aufgenommen und in StartTimer gesetzt habe, lasse ich den Film in MovieClip `timer` zwischen Frame 2 und 3 loopen. In Frame 3 kontrolliere ich jedes Mal, ob die Differenz der aktuellen Laufzeit und der Laufzeit, die ich in Frame 1 in StartTimer zwischengespeichert habe, größer oder gleich 999 ms ist. Wenn dieses der Fall ist, bedeutet das, dass fast bzw. mehr als eine Sekunde (999 ms) vorbei ist.

Dann gebe ich den `Play`-Befehl für die Ebene, die unter MovieClip `timer` liegt, also MovieClip `time`.

Nun, hier stellt sich vielleicht für einige die Frage, warum ich den Ablauf einer Sekunde gerade so kontrolliere. Ich habe im Verlauf dieser Problematik mehrere Varianten getestet, um den Ablauf einer Sekunde so genau wie eben möglich gewährleisten zu können. Zu den Varianten gehört unter anderem die klassische Framemöglichkeit: Ich stelle den Film auf 20 Bilder/Sekunde, und nach 20 Frames wäre dann eine Sekunde vorbei. Diese Variante erwies sich als ungenau, weil ich eben nicht mit jedem System gewährleisten kann, dass nach 20 Frames auch tatsächlich exakt eine Sekunde vorbei ist. Um ein Beispiel zu erwähnen, die ersten Plug-in-Versionen von Flash 5 hatten einen Bug, der dafür sorgte, dass die Filme schneller abliefen als in der vierten Version.

Abbildung 11 ▶
Frame 3 von MovieClip time

Auch der Versuch, mit einem Streamingsound die eingestellten Bilder pro Sekunden exakt abspielen zu lassen, stellte sich als ungenau heraus.

Die oben aufgeführte Variante war bei hoch eingestellter Bildrate definitiv die genaueste Möglichkeit, den Ablauf einer Sekunde kontrollieren zu können. Wenn ich beispielsweise die Bildrate auf 120 Frames pro Sekunde einstelle, so kann ich davon ausgehen, dass der Ablauf der Sekunde ca. 60 Mal in der Sekunde kontrolliert wird.

In Frame 3 von MovieClip `time` sehen wir, dass MovieClip `timer` nicht mehr präsent ist. Dies garantiert mir, dass `StartTimer` erneut gesetzt wird, wenn ich in MovieClip `time` wieder in Frame 2 springe.

Wie oben beschrieben springe ich in Frame 3, wenn eine Sekunde abgelaufen ist. Nun setze ich die Variable `s_i` neu, indem ich sie durch `s_i + 1` ersetze. Dies hat zur Folge, dass `s_i` bei jedem Durchlauf in Frame 3 um eins erhöht wird.

Nachdem ich `s_i` aktualisiert habe, kontrolliere ich, ob der Wert von `s_i = 60` ist, d.h. eine Minute vorbei ist. Falls das eintrifft, setze ich `s_i` auf null, um den Beginn einer neuen Minute starten zu können. Des Weiteren lasse ich den Film durch `Go to and Play (_currentframe+1)` in Frame 4 springen.

Wenn `s_i` nicht 60 entspricht, wird der ELSE-Part der IF-Anweisung durchgeführt. Dies bewirkt, dass ich durch `Go to Previous Frame` zurück in den Frame 2 springe, wo ja wieder der MovieClip `timer` liegt und folglich wieder der Ablauf einer Sekunde abgewartet wird, um anschließend wieder in den Frame 3 zu springen.

Wie man sehen kann, hat dieses zur Folge, dass der Film eine Minute lang zwischen Frame 2 und 3 im MovieClip `time` gefangen wird.

In Abbildung 12 sehen wir, was passiert, wenn nun eben die besagte Minute vorbei ist, also die IF-Anweisung von Frame 3 mich in diesen Frame 4 gebracht hat.

◀ **Abbildung 12**
Frame 4 im MovieClip time

Ich setze die Statusvariable, die auf der Grundebene liegt, mit `Set Variable ../:status = 0` auf null zurück. Anschließend führe ich erneut durch die `GetURL`-Anweisung die JavaScript-Funktion `gib_mir()` aus und lasse die bisherigen Variablen `hours`, `minutes`, `seconds` etc. überschreiben durch die Daten, die `gib_mir` zurückgibt. Und siehe da, die Daten sind synchronisiert.

Frame 5 ist nur noch Formsache. Hier wird lediglich kontrolliert, ob die Statusvariable gesetzt wurde, um dann in Frame 1 von MovieClip `time` (Label:start) erneut fortfahren zu können.

Gratulation!!! Sie haben jetzt, zumindest was die Flash 4-Varianten angeht, das Gröbste überstanden!

Die nun folgende Fla datum.fla, die in ohne_syn.html eingebettet ist, unterscheidet sich technisch eigentlich nur in der Zeitberechnung für die laufende Uhrzeit. Dadurch, dass hier die Variablen nicht synchronisiert werden und über den URL- beziehungsweise Source-Aufruf übergeben werden, macht dies die Sache also nur noch leichter.

Man wird auch feststellen, dass die FLAs für die Perl- und PHP-Variante fast identisch sind mit der soeben beschriebenden FLA. Die Unterschiede, die sich überwiegend auf das Einlesen der Daten beziehen, sind auch hier nicht gravierend.

Also packen wir es an und freuen uns schon auf die Flash 5-Variante, die am Ende des Tutorials etwas Neues bringen wird.

datum.fla

Der Unterschied zwischen datum_sy.fla und dem datum.fla ist auf den ersten Blick zu erkennen. Wie man sieht, gibt es in der Grundebene nur einen Frame, d. h. es kann auf einen Preloader verzichtet werden. Es gibt auch keinerlei ActionScript, welches das Einlesen der Daten bewirken würde. Logisch, da diese ja bereits mit dem URL-Aufruf übergeben wurden.

Das hier in Frame 1 von MovieClip `time` zu sehende ActionScript sollte uns aus Abbildung 6 bekannt sein. Hier werden die übertragenen Variab-

Abbildung 13 ▸
Frame 1 der Grundebene von datum.fla

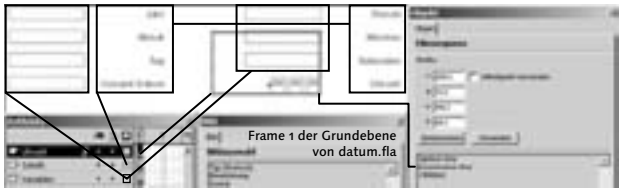


Abbildung 14 ▸
Frame 1 von MovieClip time von datum.fla

lenwerte von `hours, minutes und seconds` an die internen Variablen `st_i`, `m_i` und `s_i` übergeben.

In Frame 2 von MovieClip `time` sehen wir wieder das Prinzip, mit MovieClip `timer` den Ablauf einer Sekunde zu kontrollieren.

Des Weiteren setzen wir hier die Positionen für die Zeiger der Analoguhr und machen auch hier die kosmetische Überarbeitung der Variablen `st`, `m` und `s`. All dieses funktioniert auf der gleichen Basis, wie ich es im Film »datum_sy« beschrieben habe.In Frame 3 von MovieClip `time` sehen wir nun den gravierendsten Unterschied zu datum_sy und den folgenden Perl- und PHP-FLAs.

Da wir in dieser FLA (datum.fla) keine Synchronisation haben, müssen wir also die Funktionsweise einer Uhr in ActionScript programmieren. Hierfür betrachten wir uns die IF-Anweisungen, zu denen wir gelangen, nachdem eine Sekunde vorüber ist, etwas genauer. Um uns kurz zu erinnern, der Film wird auf Frame 2 gestoppt, und MovieClip `timer` bewirkt nach Ablauf einer Sekunde, dass der Film wieder abgespielt wird und folglich in Frame 3 gelangt, wo sich eben die besagten IF-Anweisungen befinden.

Der Sekundenwert wird um 1 erhöht (es ist ja auch eine Sekunde vergangen).

Wenn nun der Sekundenwert = 60 ist, bedeutet dies, dass eine Minute vorbei ist. Also setzen wir den Wert der Sekunde zurück auf 0 und den

◀ **Abbildung 15**
Frame 3 von MovieClip time

Minutenwert um 1 nach oben. Jetzt müssen wir natürlich prüfen, ob der Minutenwert = 60 ist, also eine Stunde vorbei ist. Falls das eintrifft, müssen wir, äquivalent zur Sekunde, den Minutenwert auf 0 zurücksetzen und den Stundenwert um 1 erhöhen. Jetzt müssen wir selbstverständlich eruieren, ob der Tag vorbei ist, also der Stundenwert = 24 ist. Falls sich dies bestätigt, müssen wir den Stundenwert auf null setzen, und der Tag beginnt von neuem.

Mit `Go to Previous Frame` gehen wir zurück auf Frame 2 und bewirken somit, dass eine Schleife entsteht, welche die laufende Uhrzeit erst ermöglicht.

Simsalabim, dies ist die Umsetzung der Funktion einer Uhr in ActionScript.

```
Set Variable: "s_i" = s_i+1
If (s_i = 60)
   Set Variable: "s_i" = 0
   Set Variable: "m_i" = m_i+1
   If (m_i = 60)
      Set Variable: "m_i" = 0
      Set Variable: "st_i" = st_i+1
      If (st_i = 24)
         Set Variable: "st_i" = 0
      End If
   End If
End If
Go to Previous Frame
```

Einen Nachteil hat diese Variante allerdings immer noch. Wenn ein Tag vorüber ist, wird das Datum nicht aktualisiert. Ich habe dieses hier nicht berücksichtigt, weil ich mit dieser Programmierung schon wesentlich weiter gegangen bin als der Großteil der Web-Developer für Mac Internet Explorer gegangen wäre. Das ist auch berechtigt, wenn wir uns einmal die Statistik von Mac-Usern im Netz betrachten (http://www.webhits.de/webhits/inetstat.htm). Diese besagt nämlich, dass die Verbreitung von Mac-Usern im Netz bei 3,2 % liegt, und bedenkt man, dass Mac-User prinzipiell eine Abneigung gegenüber Microsoft-Produkten haben, wird diese Variante wohl sehr selten in Aktion treten müssen.

Ein letztes Mal: Ich glaube, ich habe es in diesem Tutorial mehr als oft genug erwähnt, dass diese Variante keine Synchronisation erlaubt und dass je nach System, Prozessor, Plug-in-Version und Browser die Minute bis zu vier Sekunden verlieren kann. Also bitte, verwenden Sie diese Varianten nicht, um eine exakte Zeitdarstellung zu gewährleisten.

Online-Beispiel
Ein weiteres interessantes Online-Beispiel finden Sie unter http://www.nettrek.de/f5prof/js/

Die Perl-Variante

Das Skript

Ich möchte mich bei diesem Skript und dem folgenden PHP-Script wesentlich kürzer halten. Im Prinzip machen diese Skripten nichts anderes als die JavaScript-Funktion `gib_mir()`. Wesentliche Unterschiede werde ich näher erläutern.

Perl
Um Details zu einzelnen Perl-Ausdrücken, Befehlen und/oder Syntax zu erfahren, empfehle ich, eine der folgenden Sites zu besuchen:
http://www.phy.uni-bayreuth.de/~btpa25/perl/perl_main.html
http://www.perl-center.de/
http://www.perl.com/pub
http://www.builder.com/Programming/CGI/?st.bl.prog.feat

```
01 #!/usr/bin/perl -w
02
03 use strict;
04 use Time::localtime;
05
06 my $tm=localtime;
07 my $sekunde=$tm->sec;
08 my $minute=$tm->min;
09 my $stunde=$tm->hour;
10 my $tag=$tm->mday;
11 my $monat=($tm->mon);
12 my $jahr=($tm->year+1900);
13 my $jahr4=($tm->year+1900);
14 my $fertig = 1 ;
15
16 $monat++ ;
17 $tag  = "0$tag"   if ($tag  < 10) ;
18 $monat = "0$monat"  if ($monat < 10) ;
19 $jahr  = substr($jahr,2,2) ;
20
21 print "Content-Type: text/html\n\n" ;
22 print "gesamt_datum=$tag.$monat.$jahr&year=$jahr4&mon=
   $monat&day=$tag&hours=$stunde&minutes=$minute&seconds=
   $sekunde&status=$fertig";
```

Zeile 01: Die Zeile beginnt mit »#«, was von Perl als Kommentar angesehen und folglich ignoriert wird. Des Weiteren wird hier der Shell mitgeteilt, dass es sich hier um ein Perl- und nicht um ein Shell-Skript handelt. Dabei gebe ich den Installationsort von Perl auf dem Serverrechner an. Wichtig: Wenn dieses Skript von Ihnen übernommen wird, fragen Sie Ihren Provider oder den Webadministrator nach dem besagten Pfad. Diese Zeile muss eventuell modifiziert werden. Die Option »-w« ist eine Art Debugger und führt bei Skriptfehlern zu detaillierten Fehlermeldungen, die in der Error.log-Datei auf dem Server abgelegt werden.

Zeile 03–04: Hier wird durch USE: Programmcode aus externen Dateien eingebunden und beim Kompilieren fest in das Skript implementiert. Dabei sind `strict` und `time` Standardmodule von Perl, und der Ausdruck `Time::localtime` bewirkt, dass lediglich die Funktion `localtime` aus dem Modul Time verwendet wird. Man stelle sich Module als Funktionssammlungen vor.

Zeile 06–14: Hier wird ein Block von lokalen Variablen über `"my"` definiert.

Zeile 06: Hier wird ein Objekt mit Name `tm` definiert, das alle Daten der Uhrzeit und des Datums beinhaltet. Dabei ist `tm` eine Art »Array«. Analog siehe JavaScript-Erläuterung für Zeile 011.

Zeile 07: Der Variable `sekunde` wird der aktuelle Sekundenwert zugeordnet. Dies wird durch den Ausdruck `$tm->sec` gewährleistet, der zu interpretieren ist wie: Nehme aus dem Array tm den Sekundenwert. Vergleichbar mit der JavaScript-Funktion `getSeconds`.

Zeile 08: Der Variable `minute` wird der aktuelle Minutenwert zugeordnet. Vergleichbar mit der JavaScript-Funktion `getMinutes`.

Zeile 09: Der Variable `stunde` wird der aktuelle Stundenwert zugeordnet. Vergleichbar mit der JavaScript-Funktion: `getHours`.

Zeile 10: Der Variable `tag` wird der aktuelle Tag zugeordnet. Vergleichbar mit der JavaScript-Funktion: `getDate`.

Zeile 11: Der Variable `monat` wird der aktuelle Monat zugeordnet. Vergleichbar mit der JavaScript-Funktion: `getMonth`.

Zeile 12 und 13: Den Variablen `jahr` und `jahr4` wird die aktuelle Jahreszahl zugeordnet. Vergleichbar mit der JavaScript-Funktion: `getYear`. Wir haben hier zwei Variablen für die Jahreszahl, weil wir auf `jahr` später ein substr (vergleichbar mit JavaScript-Funktion Substring) anwenden werden, um die Jahreszahl in Jahrzehnten zu erhalten. Der Ausdruck »+1900« ist zu verstehen wie bei der JavaScript-Variante siehe JavaScript-Erläuterung für Zeile 015–018.

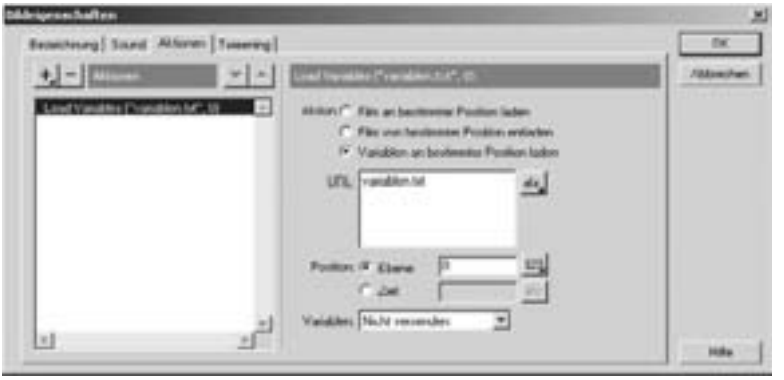
Zeile 14: Die Variable `fertig` wird mit dem Wert 1 definiert, damit ich später eine Statusvariable an Flash übergeben kann, die den Wert von `fertig` hat. Siehe JavaScript-Erläuterung für Zeile 035.

Zeile 16–18: Da Perl bei `" ->mon"` ein Wert von 0 bis 11 ausgibt (0 = Januar und 11 = Dezember), müssen wir den Wert von Monat um 1 erhöhen, um die tatsächliche Monatszahl zu erhalten. Analog: Siehe JavaScript-Erläuterung für Zeile 015 – 018.

In den darauf folgenden Zeilen steht die »kosmetische« Behandlung von Monats- und Tageswert. Wir erinnern uns an die JavaScript-Variante, wo wir das äquivalent ausgeführt haben, um vor die Werte < 10 eine 0 zu setzen. Die kryptische Anweisung: `$tag = "0$tag" if ($tag < 10) ;` bedeutet nichts anderes als `if (tag < 10).`

Wenn der Variablenwert von `tag` kleiner ist als 10,

```
tag = "0" + tag
```



◀ **Abbildung 16**
Einstellungsmöglichkeiten von Load Variables

dann ist der neue Wert von `tag` der String »0« und `tag`

```
end if
```

Ansonsten wird der Wert von `tag` nicht verändert, z. B.: tag = 9, daraus folgt: tag = »09«

Zeile 19: Der zuvor erwähnte `substr` wird auf die Variable `jahr` ausgeführt und bewirkt, dass `jahr`, z. B. für 2001, den Wert 01 erhält.

Zeile 21 – 22: Hier werden nun die berechneten Daten ausgegeben.

Um das plausibel erläutern zu können, will ich ausholen und den Befehl `Load Variable` in Flash erläutern.

Mit `Load Variable ("variablen.txt",0)` lade ich aus der Textdatei mit dem Namen Variablen die darin befindlichen Variablen herunter und lege sie auf _levelo (der Grundebene) ab. _levelo trage ich einfach bei der Position unter Ebene ein. Wie man sieht, könnte man hier aber auch ein Ziel eingeben und folglich die Variabeln in ein MovieClip ablegen, dessen Instanzname man in Ziel eingetragen hat. Ist hier aber nicht relevant.

Die Syntax einer Textdatei, die Variablen enthält, ist denkbar einfach:

```
variable=wert&variable2=wert2&variable3=wert3
```

Man trägt einfach den Variablennamen ein und anschließend deren Wert, wobei man dazwischen ein Gleichheitszeichen setzt. Hat man mehrere Variablen, so trennt man diese durch ein Und-Zeichen.

Wenn nun im obigen Beispiel die variablen.txt wie folgt aussehen würde:

```
name=Saban&nachname=Ünlü
```

so würde durch das `Load Variable` das Gleiche bewirkt werden, als wenn ich auf der Grundebene 2 `Set Variable` gesetzt hätte:

```
Set Variable "name" =  "Saban"
Set Variable "nachname" =  "Ünlü"
```

Nun, wo wir die Funktionsweise von `Load Variable` kennen, müssen wir also in Zeile 21 – 22 des Perl-Scripts eine Syntax schreiben, die der einer variablen.txt angepasst ist und wo die errechneten Werte den entsprechenden Variablennamen zugeordnet werden. D.h. an Stelle der Textdatei werden wir uns die Variablen aus der Ausgabe des Perl-Scripts entnehmen. Diese Ausgabe erreichen wir mit `print`. Dieser Perl-Befehl kann aus einem Skript eine Ausgabe bewirken, z.B.

```
print "Content-Type: text/html\n\n" ;
print  "Hallo";
```

Wenn ich dieses nun ausführen würde, bekäme ich »Hallo« als Ausgabe.

Mit `print "Content-Type: text/html";` definiere ich den Typ der Ausgaben in diesem Falle (`"text/html"`) der Typ: Text bzw. HTML. Wenn wir uns das Perl-Skript einmal separat anschauen, sehen wir dessen Ausgabe.

Hierfür gibt man im Browser http://www.nettrek.de/cgi-bin/datime.pl ein und erhält folgenden Ausdruck:

```
gesamt_datum=06.10.00&year=2000&mon=10&day=06&hours=15&minutes=
54&seconds=11&status=1
```

(In diesem Beispiel ist es der 06.10.00, und wir haben 15:54:11 Uhr.)

Unverkennbar ist die oben beschriebene Variablensyntax wieder zu sehen. Dies wurde bewirkt durch die Zeile 22. Hier wurde `print "gesamt_datum=$tag.$monat.$jahr&year=$jahr4 ...`

In diesem Ausschnitt sehen wir, dass die Variable `gesamt_datum` durch die zuvor errechneten Werte `$tag.$monat.$jahr` definiert wird. Die Punkte zwischen `$tag`, `$monat` und `$jahr` dienen hier nur zur Formatierung des Datums (siehe Ausdruck, den man erhält, wenn man das Skript

im Browser ausführt). Nach `gesamt_datum` wird die Variable `year` definiert durch die Skriptvariable `$jahr4`, und äquivalent dazu werden alle weiteren Variablen definiert.

Die Flash-Programmierung

Die Flash-Programmierung ist fast identisch mit der Programmierung für die JavaScript-Variante mit Synchronisation. Wesentliche Unterschiede sind folgende:

In Frame 1 der Grundebene laden wir die Variablen aus dem Perl-Script ein. Dieses bewirken wir mit dem Ausdruck:

```
Load Variables (http://www.nettrek.de/cgi-bin/datime.pl?
"&CacheBug, 0)
```

Details werde ich gleich, bei Bild 20, erläutern. Des Weiteren wird in Frame 3 die Variable `status` überprüft, und der Film wird erst in Frame 4 weitergeleitet, wenn `status = 1` ist. Ansonsten wird ein Sprung und `Play` in Frame 2 bewirkt. Dadurch kontrolliere ich, ob das Perl-Skript ordnungsgemäß durchgeführt wurde und folglich alle Variablen an Flash übermittelt wurden.

Bei Frame 4 ist ein `Stop` gesetzt, und die Bühne ist absolut identisch mit der von Bild 04.

Zurück zum Aufruf des Perl-Skriptes: Im Verlauf der Testphase stellte ich fest, dass der Internet Explorer in Kombination mit einem Proxy-Server folgendes Problem verursachte. Nur beim ersten Aufruf des Perl-Skriptes wurden dessen Funktionen ausgeführt und die aktuellen Daten übermittelt. Beim erneuten Aufruf des Skriptes wurde jedoch nicht das Skript ausgeführt, sondern lediglich die beim ersten Aufruf errechneten Daten übertragen. Dies liegt daran, dass sich der Explorer das Skript aus dem Cachee holt, es aber nicht von neuem ausführt, folglich werden die Daten nicht synchronisiert. Dieses Problem habe ich gelöst, indem ich beim Aufruf eine Variablenübergabe an das Skript simuliert habe. Diese Variable hat keinerlei Auswirkungen auf das Skript und dessen Funktion, sondern ist lediglich ein Dummy, den ich zufällig generiere und somit dem Cache einen neuen Skriptaufruf vorgaukle.

Die zu übertragende Dummy-Variable generiere ich durch:

```
Set Variable: "CacheBug" = Random (10)
```

Der Befehl **Random** (10) sorgt dafür, dass der Variabelenwert von **CacheBug** zufällig erzeugt wird und ein ganzzahliger Wert zwischen 0 und 10 ist.

URL-AUFRUF... Dieses müsste allen, die das Gesamte aufmerksam verfolgt haben, noch ein Begriff sein. In der JavaScript-Variante ohne Synchronisation habe ich diese Methode der Variablen **Übergabe** erläutert, und eben auf dieser Basis beruht auch diese Übermittlung der Variablen.

Um dieses nun in Load Variable einzuarbeiten, muss ich die URL hier als Ausdruck definieren:

```
Load Variables (http://www.nettrek.de/cgi-bin/datime.pl?"
&CacheBug, 0)
```

Wenn nun beispielsweise CacheBug den Wert 5 hätte, würde dieser Ausdruck folgendem URL-Aufruf entsprechen: http://www.nettrek.de/cgi-bin/datime.pl?5

Einfach testen und die URL im Browser eingeben, dann sieht man, dass die Dummy-Variable keinerlei Einfluss auf die Funktion des Skriptes hat und dass im Browser die aktuellen Zeitdaten ausgegeben werden.

Mit ...?5 simulieren wir die Übergabe der Variablen mit dem Namen 5 an das Skript. Dabei ist der Variablenwert von 5 nicht definiert, was hier auch nicht nötig ist.

Wenn wir die SWF jetzt erneut starten würden, ist die Wahrscheinlichkeit, dass wir die Variabel 5 wieder übermitteln, bei 1/11. Dies liegt an dem Random(10). Somit müssen wir uns keine Sorgen mehr um den Cache Bug machen.

Wem diese Sicherheit noch nicht hoch genug ist und wer »nahezu« sichergehen will, dass keine Daten aus dem Cache gelesen werden, der definiert die CacheBug-Variable einfach anders:

```
z.B.: Set Variable: "CacheBug" = Random (100) * Random (100) / (Random
(100) +1)
```

Hierbei ist die Wahrscheinlichkeit annähernd 0, dass die Daten aus dem Cache geholt werden.

In Frame 4 von MovieClip **time** sehen wir die Synchronisation der Variablen. Damit wir auch hier den Status der Datenübermittlung kontrollieren können, setzen wir die Variable **status**, die auf Grundebene liegt,

zurück auf 0. Diese wird ja auf 1 gesetzt, wenn das Skript ordnungsgemäß die Daten übermittelt hat (analog wie bei der JavaScript-Variante).

Anschließend kommt wieder der Skriptaufruf. Hier habe ich eine viel größere Wahrscheinlichkeit, dass sich der Internet Explorer die Daten irgendwann aus dem Cache holt, und folglich muss ich hier beim Aufruf eine Dummy-Variablen übermitteln, die dieses verhindert. Verwirklichen können wir dies mit:

```
Load Variables (http://www.nettrek.de/cgi-bin/datime.pl?"
&m_i&st_i, 0)
```

Wie wir sehen können, besteht die Dummy-Variable aus einer Verkettung des aktuellen Minuten- und Stundenwerts. Folglich ist erst nach 24 h das Cache-Problem »theoretisch« wieder möglich.

Online-Beispiel
Ein Online-Beispiel finden Sie unter http://www.nettrek.de/f5prof/cgi

Die PHP-Variante

Das Skript

Das PHP-Script beruht auf demselben Prinzip wie dem im Perl-Script oder in der JavaScript-Funktion `gib_mir()`. Der wesentlichste Unterschied hier ist, dass ich die benötigten Informationen nicht aus einem Objekt (Array) entnehme, sonder über die PHP-eigene Funktion `date()`. Diese gibt mir je nach Parameter die entsprechenden Daten zurück.

PHP
Details zu einzelnen PHP-Befehlen, der Syntax, den Funktionen etc. können Sie sehr gut bei http://www.php-center.de/ nachschlagen. Des Weiteren findet man hier ein Online-Manual: http://www.php-center.de/de-html-manual/ (zum Teil deutsch).

```
01 <?php
02    $tag=date("j");
03    $monat=date("m");
04   $jahr4=date("Y");
05   $jahr2=date("y");
06   $ges_datum="$tag.$monat.$jahr2";
07   $stunde=date("H");
08   $minute=date("i");
09   $sekunde=date("s");
10   $fertig=1;
11   echo "year=$jahr4&mon=$monat&day=$tag&gesamt_datum=$ges_datum
     &hours=$stunde&minutes=$minute&seconds=$sekunde&status=$fertig";
12 ?>
```

Zeile 1 und 12: mit `<?php` starte ich ein PHP-Script und mit `?>` beende ich es. Kaum zu verkennen ist die Ähnlichkeit mit JavaScript.

Zeile 2: `date("j")` gibt mir die Tageszahl, die ich als Variablenwert von `tag` einsetze.

Zeile 3: `date("m")` gibt mir die Monatszahl, die ich als Variablenwert von `monat` einsetze.

Zeile 4: `date("Y")` gibt mir die Jahreszahl vierstellig, die ich als Variablenwert von `jahr4` einsetze.

Zeile 5: `date("Y")` gibt mir die Jahreszahl zweistellig, die ich als Variablenwert von `jahr2` einsetze.

Zeile 6: Hier definiere ich die Variable `ges_datum` durch die Verkettung der bisher erhaltenen Variablen und den (».«) Punktzeichen. Daraus folgt, dass ich Tag, Monat und die zweistellige Jahreszahl im Datumsformat dd.mm.yy ausgebe. Z. B.: tag=1 monat=1 jahr=00 d. h. ges_datum = 1.1.00

Zeile 7: `date("H")` gibt mir die Stundenzahl, die ich als Variablenwert von `stunde` einsetze.

Zeile 8: `date("i")` gibt mir die Minutenzahl, die ich als Variablenwert von `minute` einsetze.

Zeile 9: `date("s")` gibt mir die Sekundenzahl, die ich als Variablenwert von `sekunde` einsetze.

Zeile 10: Hier definiere ich die Variable fertig mit dem Wert 1. Den Variablenwert übermittle ich dann mit der Variable `status` an Flash. Diese wird in Flash geprüft und erlaubt es so, dass kontrolliert werden kann, ob das Skript ordnungsgemäß durchgeführt wurde. (Das Prinzip habe ich zuvor bereits erläutert.)

Zeile 11: Die zuvor bestimmten Daten werden nun an Flash weitergegeben. Äquivalent habe ich dies bei CGI-Skript gemacht und ausführlich erläutert (siehe Zeile 21–22 der Perl-Erläuterung). Der einzige Unterschied ist der, dass bei PHP der Befehl `echo` und bei Perl der `print`-Befehl benutzt wurde. Die Befehle selbst unterscheiden sich von der Funktionsweise allerdings nicht wesentlich!

Die Flash-Programmierung

Ist hier fast absolut identisch mit der FLA für das Perl-Skript. Der einzige Unterschied hier ist, dass bei `Load Variable` eine andere URL eingegeben werden muss! Diese verweist dann auf das PHP 3-Skript, das im selben Ordner liegt wie die SWF.

Dieses gilt verständlicherweise sowohl für die URL am Anfang der FLA sowie für die URL der Synchronisation.

Analog zu Bild 19 (Frame 1 der Grundebene) ist die URL folglich:

```
Load Variables ("datime.php3?"&CacheBug, 0)
```

Und zu Bild 21 (Frame 4 von MovieClip `time`):

```
Load Variables ("datime.php3?"&m_i&st_i, 0)
```

Online-Beispiel
Das Online-Beispiel finden Sie unter http://www.nettrek.de/f5prof/php/

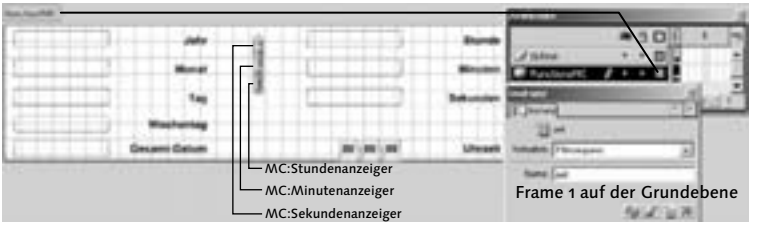
Die Flash 5-Variante

Die Flash-Programmierung

Nun haben wir es geschafft! Wir kommen jetzt zu dem Teil des Tutorials, in dem wir mit Flash 5 auf die ganze JavaScript-, Perl-, PHP- etc. Geschichte verzichten können.

Im Flash 5 gibt es jetzt eigene Datumsfunktionen, die es mir ermöglichen, dass ich die Ermittlung aller benötigten Informationen intern abwickeln kann.

Abbildung 17 ▶
Frame 1 der Grundebene



Ich kann aber auch hier auf eine Art Skript nicht verzichten. Dieses lege ich in ein MovieClip und kann, ähnlich wie mit »Call«, darauf zugreifen. Das Skript besteht aus mehreren Funktionen, und ich meine wirklich Funktionen. Es ist nun möglich, diese in Flash 5 zu erstellen und aus dem gesamten Film darauf zuzugreifen.

In Frame 1 der Grundebene von datum.swf sehen wir ein eigentlich bekanntes Bild! Optisch sind nur Unterschiede dadurch zu erkennen, dass wir ein Textfeld mehr haben und dass die Zeiger von der Analoguhr sich verändert haben.

Das Textfeld beinhaltet hier den Wochentag, d.h. die Variable: `dayname`. Diese haben wir bisher nicht benutzt, und sie hat lediglich die Aufgabe, den Wochentag (Mo bis So) wiederzugeben.

Bei Flash 5 muss man Textfelder, die dynamisch sind, d.h. Variablen beinhalten, anders als bei Flash 4 einrichten. Nachdem das Feld generiert wurde, muss man die in Abbildung 18 gezeigten Einstellungen unter TEXT OPTIONS einstellen.

Die Zeiger der Analoguhr in Frame 1 unterscheiden sich nicht nur durch ihre Form (auch hier ist der Mittelpunkt zentriert am unteren Ende), sondern auch dadurch, dass ich hier objektbezogene Programmierung anwende. Dazu kommen wir aber später.

Viel wichtiger ist hier der in Layer »FunctionMC« vorhandene MovieClip namens `zeit`: In Frame 1 von MovieClip `zeit` definieren wir die zuvor erwähnten Funktionen.

Allgemein zur **function**-Syntax: Die Ähnlichkeit zu JavaScript ist eindeutig!

```
function FUNKTIONSNAME (PARAMETER:String, Integer, etc.)
{
FUNKTIONSINHALT
}
```

◀ **Abbildung 18**
Einstellungen unter Textoptionen, die Sie vornehmen müssen

Mit Hilfe von `Funktion` kann man oft vorkommende Action-Anweisungen als Routine definieren. Mit dem `FUNKTIONSNAMEN` spricht man die Anweisungsfolge an; dabei kann man Parameter übergeben, falls dies nötig ist. Ein Parameter kann als String (in Anführungsstrichen), als Integer etc. übergeben und dann im `FUNKTIONSINHALT` weiterverarbeitet werden. Der `FUNKTIONSINHALT` sind die ActionScript-Anweisungen, die in der Routine ausgeführt werden sollen. Start und Ende einer Funktion definiert man analog zu JavaScript über `{` und `}`.

```
001      function WochentagArray () {
002      this[0] = "So";
003      this[1] = "Mo";
004      this[2] = "Di";
005      this[3] = "Mi";
006      this[4] = "Do";
007      this[5] = "Fr";
008      this[6] = "Sa";
009      return (this);
010   }
011
012   function zweiStelligJahr (year) {
013      shortyear = year%100;
014      if (shortyear<10) {
015         shortyear = "0"+shortyear;
016      }
017      return shortyear;
018   }
019
```

```
020  function vierStelligJahr (year) {
021     if (year>1900) {
022        return year;
023     }
024     return year+1900;
025  }
026
027  function Datumsdaten (was_brauchst) {
028     Wtag = new WochentagArray();
029     datum = new Date();
030     tag = datum.getDate();
031     monat = datum.getMonth();
032     jahr = datum.getYear();
033     if (was_brauchst == "derTag") {
034        if (tag < 10) {
035           tag = "0"+tag;
036        }
037        return tag;
038     }
039     if (was_brauchst == "derWtag") {
040        welcherTag = Wtag[datum.getDay()];
041        return welcherTag;
042     }
043     if (was_brauchst == "derMonat") {
045        if (monat < 10) {
046           monat = "0"+monat;
047        }
048        return monat;
049     }
050     if (was_brauchst == "dieJzahl2") {
051        return zweiStelligJahr(jahr);
052     }
053     if (was_brauchst == "dieJzahl4") {
054        return vierStelligJahr(jahr);
055     }
056     if (was_brauchst == "gesamteDatum") {
057        if (monat < 10) {
058           monat = "0"+monat;
059        }
060        if (tag < 10) {
```

```
061            tag = "0"+tag;
062         }
063         datum_ges = Wtag[datum.getDay()]+" der "+tag+"."+monat+"
."+zweiStelligJahr(jahr);return datum_ges;
         }
064   }
065
066   function zeitfunktion (was_brauchst) {
067      uhrzeit = new Date();
068      if (was_brauchst == "dieSekunde") {
069         sekunde = uhrzeit.getSeconds();
070         return sekunde;
071      } else if (was_brauchst == "dieMinute") {
072         minute = uhrzeit.getMinutes();
073         return minute;
074      } else if (was_brauchst == "dieStunde") {
075         stunde = uhrzeit.getHours();
076         return stunde;
077      }
078   }
```

Zeile 001–010: Hier definiere ich eine Funktion, die dynamisch ein Array() erzeugt. Später kann man diese Funktion benutzen, um über `"new"` ein Objekt zu generieren, das den im `FUNKTIONSINHALT` definierten Array beinhaltet.

Z. B. `Wochentag = new WochentagArray`

d. h. Wochentag [0] = »Sonntag«, Wochentag [1] = »Montag« usw.

Der Vorteil eines Funktionsarrays gegenüber einem klassisch erzeugten Array ist, dass ich bei mehreren identischen Arrays, z. B. `ZielWochentag` und `WunschWochentag`, viel Schreibarbeit sparen kann.

Zeile 012–018: Der Funktion `zweiStelligJahr` muss beim Aufruf ein Parameter (`year`) als Integer übergeben werden. Damit macht man dann Folgendes:

Zeile 013: Hier wird die lokale Variable `shortyear` definiert als `year%100`.

% ist eine mathematische Funktion von Flash, die den Restwert einer Division zurückgibt. Die Syntax ist: Ergebnis = Ausgangswert%Divisor,

d.h. wenn z.B. in unserem Falle `year` 2001 wäre, so würde das Ergebnis (also: shortyear) 1 sein, weil 2001:100 ist 20, und der Rest ist 1.

Zeile 014–016: Ist die altbekannte »kosmetische« IF-Anweisung, die bei Werten kleiner 10 einen String zurückgibt, der zusammengesetzt ist aus dem Zeichen »0« und dem Wert.

Zeile 017: Durch Return kann man aus einer Funktion eine »Ausgabe« erzwingen. In diesem Falle wird die errechnete lokale Variable `shortyear` zurückgegeben.

Zeile 020–025: Die Entstehungsgeschichte der Funktion `vierStelligJahr` beruht auf der späteren Verwendung von der Flash-Funktion `getYear`. `getYear` bewirkt, dass Flash die Jahreszahl des Clientrechners ausgibt. Der Ausdruck von `getYear` ist äquivalent zu dem, den die JavaScript-Funktion `getYear` ausgibt, d.h. der aktuellen Jahreszahl wird 1900 entnommen,

z.B. 2000 => Ausgabe 100

Die IF-Anweisung in Zeile 021–023 ist hier also theoretisch unnötig und dient lediglich der Sicherheit, dass keine falsche Daten an die Funktion übermittelt werden. Entscheidender ist hier die Zeile 024: `return year+1900`.

Hier gibt die Funktion also die vierstellige Jahreszahl dadurch aus, dass sie die Subtraktion von `getYear` wieder rückgängig macht.

Zeile 027 – 064: Hier wird die tragende Funktion für die Datumsdaten definiert: Datumsdaten. Wichtig bei Aufruf dieser Funktion ist die Übergabe des Parameters `was_brauchst` in Form eines Strings. Je nachdem, welcher Parameter übergeben wurde, gibt Datumsdaten die entsprechende Ausgabe.

Parameter	Ausgabe	Definiert in Zeile (von – bis)
»derTag«	Tageszahl	033 – 038
»derWtag«	Wochentag [So ... Mo]	039 – 042
»derMonat«	Monatstag	043 – 049
»dieJzahl2«	zweistellige Jahreszahl	050 – 052
»dieJzahl4«	vierstellige Jahreszahl	053 – 055
»gesamteDatum«	Gesamtes WT der dd.mm.yy	056 – 063

Am Anfang dieser Funktion definiere ich die im Folgenden benötigten lokalen Variablen.

Zeile 028: Hier wird mit Hilfe der zuvor definierten Funktion `WochentagArray` ein neues Objekt namens `Wtag` generiert, das dem Array `Wtag[0]="So" ...Wtag[6]="Mo"` entspricht.

Zeile 029: Hier definiere ich das Objekt `datum` durch die Flash-Funktion `Date()`. Die Funktionsweise ist vergleichbar mit der zuvor erläuterten JavaScript-Funktion `Date()`, siehe JavaScript-Erläuterung für Zeile 011. Es wird quasi ein `Array (:datum)` definiert, der alle Datumsinformationen innehat.

Zeile 030: Hier definiere ich die Variable `tag` durch `datum.getDate();` die Syntax bedeutet nichts anderes, als dass ich aus dem Objekt `datum` über `getDate()` die Tageszahl entnehme.

Zeile 031: Äquivalent zu Zeile 030, nur dass hier über `getMonth()` die Monatszahl als Variablenwert von `monat` definiert wird.

Zeile 032: Äquivalent zu Zeile 030, nur dass hier über `getYear` die Jahreszahl als Variablenwert von Jahr definiert wird. Siehe auch Zeile 020-025.

Zeile 033–038: Dieser Block wird aktiv, wenn der übergebene Parameter von Datumsdaten exakt `( == ) "derTag"` ist. Diese Bedingung wird in Zeile 033 durch die IF-Anweisung kontrolliert.

Der Block selbst beinhaltet wieder eine IF-Anweisung, und sie ist uns allen bekannt als ... »kosmetische« IF-Anweisung. Hier kontrolliere ich, ob die lokale Variable `tag` kleiner 10 ist, um dann folglich eine 0 davor zu setzen. Durch die in Zeile 037 definierte Return-Anweisung wird dann der Variablenwert von `tag` zurückgegeben.

Zeile 039–042: Dieser Block wird ausgeführt, wenn der Parameter von Datumsdaten `== "derWtag"` ist. Hier wird dann die Variable `welcherTag` definiert aus der Kombination des Objektes `Wtag` und der Funktion `getDay()`.

Durch `datum.getDay()` erhalte ich aus dem Objekt `datum` die Ausgabe des Wochentages. Dabei ist die Ausgabe der Flash-Funktion `getDay()` definiert als ganzzahliger Wert von 0 bis 6 (So = 0 ... Mo = 6). Diese Ausgabe setze ich in den Array `Wtag` ein, so dass, z. B. auf einem Montag, der Ausdruck `Wtag[datum.getDay]` identisch ist mit: `Wtag[6]`. Laut Objektdefinition der Funktion `WochentagArray` ist dann `Wtag[6] = "Mo"` und folglich der Variablenwert von `welcherTag = "Mo"`.

Dieser wird anschließend durch die Return-Anweisung in Zeile 041 zurückgegeben.

Zeile 043–049: Dieser Block beinhaltet ebenfalls eine »kosmetische« IF-Anweisung bezogen auf die Monatszahl und folglich auf die Variable `monat`. Dieser Abschnitt wird ausgeführt, wenn der Parameter von Da-

tumsdaten == "derMonat" ist, und er gibt den Variablenwert von monat zurück. Äquivalent zu "derTag".

Zeile 050 – 052: Dieser Abschnitt wird aktiv, wenn der Parameter von Datumsdaten == "dieJzahl2" ist. Hier wird dann in Zeile 051 der Wert zurückgegeben, den man durch die Ausgabe der Funktion zweiStellig-Jahr(jahr) erhält. Wie man sieht, wird dabei der Funktionsparameter jahr, d.h. die soeben definierte Jahreszahl, an zweiStelligJahr übergeben.

Zeile 053 – 055: Ist Äquivalent zur Zeile 050 – 052; nur dass hier die vierstellige Jahreszahl durch die Funktion vierStelligJahr(jahr); zurückgegeben wird, wenn der Parameter von Datumsdaten == "dieJzahl4" ist.

Zeile 056 – 063: Dieser Abschnitt wir ausgeführt, wenn der Parameter von Datumsdaten =="gesamteDatum" ist. Hier wird dann der Variablen datum_ges eine Zeichenkette zugeordnet, die aus den bisher ermittelten Datumsinformationen besteht. Dieser String wird als Verknüpfung in Zeile 63 wie folgt zusammengesetzt:

```
datum_ges = Wtag[datum.getDay()]+" der "+tag+"."+monat+"."
+zweiStelligJahr(jahr);
```

d.h. Wtag[datum.getDay()] (Der ermittelte Wochentag) +" der " (Die Zeichenkette » der «) +tag (Die Tageszahl) + "." (Das Punktzeichen) + monat (Die Monatszahl) + "." (Das Punktzeichen) + zweiStelligJahr(jahr) (Die zweistellige Jahreszahl)

Als Beispiel könnten wir uns den Wert von datum_ges für den 18.10.00 anschauen, das wäre dann: Mi der 18.10.00

In Zeile 64 wird dieser Wert durch Return dann ausgegeben.

Zeile 066–078: Definition der Funktion namens zeitfunktion.

Im späteren Funktionsaufruf ist auch hier die Übergabe eines Parameters erforderlich, damit eine entsprechende Ausgabe möglich wird. Die Funktion selbst beginnt mit der Erzeugung des Objektes uhrzeit, analog zu der Erstellung des Objektes datum zuvor.

Zeile 068–070: Ist der erste Anweisungsblock und wird ausgeführt, wenn der Parameter vom Funktionsaufruf zeitfunktion == "dieSekunde" ist. In diesem Falle wird aus dem Objekt uhrzeit der Sekundenwert durch den Ausdruck uhrzeit.getSeconds() entnommen und der Variablen sekunde zugeordnet. Anschließend erfolgt die Rückgabe des Variablenwertes sekunde in Zeile 070.

Zeile 071–073: Sollte der erste Anweisungsblock nicht ablaufen, wird überprüft, ob der Parameter von `zeitfunktion = "dieMinute"` ist, und analog zu dem zuvor erläuterten Verfahren wird die Variable `minute` durch `uhrzeit.getMinute()` definiert und anschließend ausgegeben.

Zeile 074–078: Sollte auch der zweite Block nicht ablaufen, wird kontrolliert, ob der Parameter der Funktion `zeitfunktion == "dieStunde"` ist, und analog wie zuvor beschrieben wird die Variable `stunde` zurückgegeben, die als `uhrzeit.getHours()` definiert wurde.

Diese verschachtelte IF-Anweisung sorgt dafür, dass bei einem erkannten Parameter die IF-Anweisung sofort beendet wird. Kleiner, aber nicht unwesentlicher Geschwindigkeitsvorteil, gerade weil wir diese Funktion am häufigsten benutzen werden.

Nun haben wir alle Funktionen definiert, um im Verlauf der FLA eine laufende Uhrzeit und die Ausgabe der Datumsinformationen ermöglichen zu können.

Auf der Grundebene befindet sich auch in dieser FLA die uns bekannte MovieClip `time`!

In Frame 1 von MovieClip `time` definiere ich mit Hilfe der zuvor beschriebenen Funktionen die Variablen, die ich sonst über JavaScript, PHP bzw. Perl erhalten habe.

```
_root.hours = _root.zeit.zeitfunktion("dieStunde");
```

Ich definiere auf der Grundebene die Variable `hours` so, dass ich ihr die Ausgabe zuordne, die ich von der `zeitfunktion` erhalte, wenn ich diesem den Parameter `dieStunde` übergebe.

`_root.hours` entspricht der Flahs 4-Syntax _levelo:hours

Bei `_root.zeit.zeitfunktion("dieStunde");` halte ich folgende Syntax ein, um die Funktion: `zeitfunktion` anzusprechen`: pfad.FUNKTIONSNAME(PARAMETER).` In unserem Falle ist der Pfad ja wie bekannt `_root.zeit`, da sich die MovieClip `zeit` auf der Grundebene befindet.

Analog dazu definiert man die Variable `minutes`, `seconds`, `year`, `mon`, `dayname` und `gesamt_datum`.

```
_root.minutes = _root.zeit.zeitfunktion("dieMinute");
_root.seconds = _root.zeit.zeitfunktion("dieSekunde");
_root.year = _root.zeit.Datumsdaten("dieJzahl4");
_root.mon = _root.zeit.Datumsdaten("derMonat");
_root.day = _root.zeit.Datumsdaten("derTag");
_root.dayname = _root.zeit.Datumsdaten("derWtag");
```

```
_root.gesamt_datum = _root.zeit.Datumsdaten("gesamteDatum");
```

In Abbildung 6 haben wir bereits die Aufgaben der Variablen **st_i**, **m_i** und **s_i** erläutert. Hier definieren wir äquivalent zur Flash 4-Variante eben diese Variablen.

```
st_i = _root.hours;
m_i = _root.minutes;
s_i = _root.seconds;
```

In Frame 2 von MovieClip **time** wird die kosmetische IF-Anweisung für die Variablen der laufende Uhrzeit durchgeführt. Dies ist äquivalent zu der Flash 4-Variante:

Wenn der Wert von **s_i** (Sekundenwert) kleiner 10 ist, dann definiere die Variable **s** (Variable, die bei der laufenden Uhrzeit ausgegeben wird) als »0« + Wert(s_i). Sonst setze den Variablenwert von **s_i** als Variablenwert von **s**. Analog wird mit dem Minutenwert, d.h. der Variablen **m**, verfahren. Analog wird mit dem Stundenwert, d.h. der Variablen **st**, verfahren.

```
if (Number(s_i)<10) {
    s = "0" add s_i;
} else {
    s = s_i;
}
if (Number(m_i)<10) {
    m = "0" add m_i;
} else {
    m = m_i;
}
if (Number(st_i)<10) {
    st = "0" add st_i;
} else {
    st = st_i;
}
```

Weiterhin findet man in diesem Frame den MovieClip **timer**, dessen Funktion wir ja auch bereits kennengelernt haben.

In Frame 3 von MovieClip **time** werden nun die laufende Uhrzeit und die Synchronisation umgesetzt.

Ich ordne dem Variablenwert `s_i` die aktuelle Sekundenzahl zu. Wenn dieser Wert exakt `( == ) 0` ist, dann beginnt eine neue Minute; d.h. ich benötige den aktuellen Minutenwert m_i. Falls dieser `== 0` ist, heißt das wiederum, dass eine neue Stunde begonnen hat und ich folglich die neue Stundenzahl `st_i` benötige. Wenn wiederum diese `== 0` ist, beginnt ein neuer Tag, und ich muss alle Variablen für das Datum aktualisieren. Da diese in Frame 1 definiert werden, reicht ein `gotoAndPlay(1)`.

Falls `st_i` ungleich null ist, muss man in den vorherigen Frame, um die »kosmetische« IF-Anweisung auf den neuen Minuten-, Sekunden- und Stundenwert anzuwenden.

Analog gehe in den vorherigen Frame, wenn `m_i` ungleich null ist.

Analog gehe in den vorherigen Frame, wenn `s_i` ungleich null ist.

```
s_i = _root.zeit.zeitfunktion("dieSekunde");
if (Number(s_i) == 0) {
   m_i = _root.zeit.zeitfunktion("dieMinute");
   if (Number(m_i) == 0) {
      st_i = _root.zeit.zeitfunktion("dieStunde");
      if (Number(st_i) == 0) {
         gotoAndPlay (1);
      } else {
         prevFrame ();
      }
   } else {
      prevFrame ();
   }
} else {
   prevFrame ();
}
```

Wer schon einmal mit Delphi, C++, Director etc. gearbeitet hat, kennt es schon, dass man Objekten Befehle zuordnen kann, die ausgeführt werden, wenn ein vordefiniertes Event eintritt.

Nun ist dies auch mit F5 möglich, und ich wollte es mir nicht nehmen lassen, diesbezüglich die Zeigerrotation so zu steuern.

Um dies machen zu können, klicke ich einfach auf den z.B. Stundenzeiger, und wenn ich jetzt einen Blick auf den ActionScript-Editor werfe, sehe ich, dass neben dem Register Movie Explorer jetzt Object Action steht und nicht wie sonst Frame Action. D.h. die nun folgenden Actions beziehen sich auf das Objekt »Stundenzeiger«.

```
001   onClipEvent (enterFrame) {
002      stundenrotation = ../:st_i;
003      if (stundenrotation>12) {
004         stundenrotation = stundenrotation-12;
005      }
006      if (stundenrotation<1) {
007         stundenrotation = 12;
008      }
009      grad = stundenrotation*30;
010      this._rotation = grad;
011   }
```

Zeile 001: Hier definiere ich das Event, auf welches das Objekt ActionScript reagieren soll! In diesem Falle wird die Aktion ausgeführt, wenn man in den Frame des Objektes gelangt. Es sind aber auch noch die `Events load`: wenn geladen, Mouse down, Key down etc. möglich!

Zeile 002: Hier definiere ich die lokale Variable `stundenrotation` und ordne ihr die aktuelle Stundenzahl über die Variable `st_i` zu.

Bevor ich nun im Weiteren die Rotation erläutere, hier kurz zur Theorie. Wir kennen es aus dem Mathe-Unterricht. Wenn ich ein Objekt um 360° drehe, dann bewirke ich, dass es wieder an die ursprüngliche Position zurückkehrt .

Zeile 003: Der Stundenzeiger dreht sich innerhalb eines Tages zwei Mal um seine eigene Achse, d.h. wenn es 13:00 Uhr ist, beginnt eine neue Periode. Daher hier die Abfrage, ob die Stundenzahl > 12 ist.

Zeile 004: Falls das zutrifft, subtrahiere ich diese Zahl mit 12, weil die Periode der Stunde zwölf Einheiten hat, und entsprechend ist die Zeigerposition bei 13:00 dieselbe wie bei 01:00 Uhr.

Zeile 006–008: Da die Zeigerposition bei 00:00 dieselbe ist wie bei 12:00 Uhr, sorgt die IF-Anweisung hier dafür, dass `stundenrotation` auf 12 gesetzt wird, wenn es Mitternacht ist.

Zeile 009: Hier definieren wir die Gradzahl der Rotation. Da eine gesamte Drehung 360° entspricht und dies in zwölf (Stunden) Schritten unterteilt sein muss, bedeutet das, dass jeweils 30° einer Stunde entsprechen.

Somit dürfte klar sein, warum die Multiplikation der Variable `stundenrotation` mit 30 die Variable `grad` für die Gradzahl definiert.

Zeile 010: Über `setProperty` führe ich nun die Rotation aus. Als Target definiere ich `this`, d.h. dieses Objekt, und `grad` definiert die Gradzahl der Rotation.

Bei dem Minuten- und Sekundenzeiger wird äquivalent vorgegangen! Der Unterschied liegt hier lediglich der Definition der Gradzahl. Für den Minutenzeiger bedeutet das:

```
onClipEvent (enterFrame) {
   grad = ../:m_i*6;
   this._rotation=grad;
}
```

Der Minutenzeiger dreht sich mit 60 Einheiten um seine eigene Achse, d.h.: alle 6° ist eine Minute vorbei, so dass nach 60 Minuten die 360° erreicht sind.

Folglich ist die Gradzahl definiert als die aktuelle Minutenzahl multipliziert mit 6.

Die Rotation wird dann identisch wie beim Stundenzeiger durchgeführt.

Online-Beispiel

Ein Online-Beispiel finden Sie unter http://www.nettrek.de/f5prof/f5/

Workshops und Anwendungsbeispiele

582 Effekte mit duplicate-MovieClip

586 Passwortabfragen

586 Einfache Passwortabfrage mit mehreren Usern

588 Verschlüsselungstechnik in Flash

595 Passwortabfrage mit Flash und CGI

604 Flash 5 Detection

607 Speeddetection

610 Preloader

613 Preloader mit Flash 5

615 Preloaden von externen Movies

618 Flash Print

622 Cookies anlegen und auslesen

622 Cookie anlegen und Wert setzen

623 Cookie auslesen und gespeicherten Wert ermitteln

626 Mausposition und -verfolgung in Flash 4 und 5

626 Ermitteln der Mausposition – Lösungsansätze in Flash 4
628 Verfolgen der Maus mit Objekten in Flash 4
630 Maus- bzw. Objektverfolgung in Flash 5
634 Grafiken der Maus folgen lassen

640 Dynamisches Menüsystem erstellen

640 Ablauf

Effekte mit duplicateMovieClip

In diversen Beispielen wurde bereits die Funktion `duplicateMovieClip` verwendet, aber ich finde, unter dieser Überschrift vermutet man eigentlich zunächst einmal den folgenden Effekt.

▸ Carlo Blatz

Entwickelt haben wir ihn für den Preloader des Intros der Elsa-Webseite (siehe Sitecheck Seite 311). Hierbei wird zunächst ein Textfeld animiert (dafür müssen wie immer alle Font-Outlines eingebunden sein und das Textfeld sich in einem MovieClip befinden). Diese Animation wird dann zeitversetzt dupliziert und transparenter gemacht.

Die Animation

Der MovieClip mit dem Textfeld »text« wird im ersten Schritt rotiert, skaliert und geneigt. Diese Animation heißt »transform« und wird wiederum mit diversen Farbeffekten im MovieClip `colorize` animiert. Erst diese Instanz liegt auf unserer Bühne.

Das Skript

Im ersten Keyframe setzen wir die Variablen `i` und `Alpha` auf 1 und den Inhalt der Variablen im Textfeld auf den Wert der Variablen `name` – dies ist der Inhalt unseres Textfeldes. Die gesamte Animation wird vorerst unsichtbar geschaltet.

```
i = 1;
alpha = 1;
this.colorize0.transform.text.name = name;
this.colorize0._visible = false;
```

Ein Keyframe später beginnt das eigentliche Duplizieren. Die Animation wird dupliziert und die neue Instanz erhält den Namen »Text+i«, also im ersten Durchlauf Text1. Dieser Duplicate wird auf eine Tiefe von 900–i gesetzt, also im ersten Durchlauf auf 899. Somit wird jeder neu duplizierte MovieClip immer unter dem vorherigen angeordnet, was elementar wichtig für unseren Effekt ist.

```
this.colorize0.duplicateMovieClip (["colorize"+i], 900-i);
```

Der Inhalt des Textfeldes der neu kreierten Instanz muss auch auf den Wert `name` gebracht werden – bekommt also den gleichen Inhalt.

```
this["colorize"+i].transform.text.name =  name;
```

Man kann sich diese Zeile auch sparen, indem die Variable des Textfeldes bereits »von unten geholt wird«. Der Variablenname im Textfeld könnte z. B. `_level0:name` sein, dann würde sich jede duplizierte Instanz den Inhalt dieser Variablen holen.

Die neue Instanz wird nun um `i*alpha` vergrößert – also im ersten Durchlauf nur um 1.

```
this["colorize"+i]._xscale = this["colorize"+(i-1)]._xscale +
i*alpha;
this["colorize"+i]._yscale = this["colorize"+(i-1)]._yscale +
i*alpha;
```

Die Einheit, mit der die neue Instanz jeweils um einen Schritt transparenter wird, ist in diesem Beispiel `10-i*0,2`, d. h. im ersten Durchlauf 1,8 und im zweiten 1,6 etc. Zu guter Letzt muss `i` natürlich für den nächsten Durchlauf um 1 erhöht werden.

```
this["colorize"+i]._alpha = 10-i*0.2;
i++;
```

Ein weiteres Keyframe später wird dem Loop mitgeteilt, wie oft sich der MovieClip duplizieren soll. Wenn er bereits 12 Mal gestartet ist, stoppt er und man kann die Animation bestaunen.

```
if (i<12) {
   gotoAndPlay (_currentframe-1);
} else {
   stop ();
}
```

Man kann mit wenigen Handgriffen dieses Script für ganz eigene Effekte modifizieren. Der einfachste Weg ist natürlich, die Tweenings in `Colorize` und `Transform` zu verändern. Man kann aber auch mit den Zahlen im Script spielen, dann verändert man den Effekt stark. Die vorliegenden Werte sind ein guter Kompromiss zwischen optischem Eindruck und Performance. Natürlich wäre der Effekt mit 100 Duplikaten noch schärfer, aber das werden wohl nur die wenigsten Rechner mitmachen.

Weitere Effekte mit DuplicateMovie finden Sie im Kapitel »Animation mit Typographie« ab Seite 224.

Passwortabfragen

Auch Flasher müssen manchmal etwas verbergen. An Passwortabfragen versucht man sich seit Jahren. Wir möchten Ihnen einige Lösungsansätze mit ihren Vor- und Nachteilen zeigen.

Einfache Passwortabfrage mit mehreren Usern

▶ Carlo Blatz

Eine Passwortabfrage an sich ist nicht allzu schwer zu programmieren. Will man hingegen mehrere User verwalten, wird es schon etwas kniffliger! Wir möchten unsere Daten aus einer externen Datenquelle beziehen, damit der Kunde seine User-Accounts auch selbst pflegen und verwalten kann. Von vorn:

Um die User-Daten komfortabel zu ändern, liegen die Daten in einem Textfile (bei professionellen Lösungen in einer Datenbank). Das hat auch den Vorteil, dass die Variablen nicht im SWF sind, denn dort sind sie unverschlüsselt und mit jedem Hex-Editor lesbar.

Das Textfile:

```
name1=Calle&
&pass1=Powerflasher&
&link1=http://www.powerflasher.de&
&name2=Mare&
&pass2=Gerald&
&link2=http://www.tuer3.com&
&name3=&
&pass3=&
&link3=http://www.flashworker.de&
&ende=4&
```

Auch das letzte `&` nicht vergessen, da Flash 5 sonst den Wert nicht als Number interpretiert. Solche Fehler können Schwierigkeiten bereiten.

Hier können nun beliebig viele User mit jeweils eigenem Passwort eingegeben werden. `Name3` und `pass3` ist freigelassen, das Skript springt somit zu der URL, wenn der User nichts eingibt. Damit Flash in keine End-

losschleife gerät, gibt es eine Variable `Ende`. Flash durchsucht also so viele Datensätze nach Übereinstimmungen, bis `Ende` erreicht ist. In Flash brauchen wir nun nur am Anfang die Aktion, um das Textfile zu laden (`Load Variables`), danach wird die Zeitleiste gestoppt und wartet auf die Eingabe des Users in zwei Textfeldern namens `"name"` und `"pass"`. Hat er dies getan, kann er mit dem Startknopf bestätigen (der Knopf sollte auch auf (↵) reagieren), und nun wird das Auswertungsskript gestartet. N ist ein Wert, der nach jeder Überprüfung um 1 erhöht wird, solange `n` kleiner als `ende` ist.

Mit anderen Worten:

Solange `n` kleiner ist als `ende`, überprüfe, ob `name(n)` ist gleich Benutzername und `pass(n)` ist gleich `passwort`.

Der ActionScript-Code:

```
n = 1;
while (Number(n)<ende) {
   if (eval("name" add n) eq Benutzername and eval("pass" add n) eq
passwort) {
      getURL (eval ("link" add n));
      gotoAndStop ("user1");
      n = ende;
   } else {
      n++;
      gotoAndStop ("falsch");
   }
}
```

Am Anfang wird also n auf 1 gesetzt, damit bei jedem Durchlauf wieder alle Datensätze überprüft werden. Dann kommt die Abfrage, die bei Erfolg zur benutzerspezifischen URL weiterleitet und für unseren Testlauf noch zum Frame `"user"` springt. Danach wird n dem Wert `ende` gleichgesetzt, damit die Schleife beendet wird. Wenn der User ein falsches Passwort eingibt, wird n um 1 erhöht, und die Zeitleiste springt zum Frame `"falsch"`.

Wem es egal ist, dass Flash 4-User nicht mit der Abfrage zurechtkommen, kann auch ein Flash 5-only-Script verwenden und so auf das Eval verzichten:

```
   if (this["name"+n] == Benutzername && this["pass"+n] == passwort) {
      getURL (this["link"+n]);
```

Der Rest bleibt wie gehabt.

Wir haben das Skript nun noch dahingehend erweitert, dass der User nur drei Mal falsch eingeben kann. Im Frame »falsch« springt der Zurück-Button nur drei Mal zurück. Bei einem weiteren Versuch springt er zum Frame »out«. Das wird mit der Variable **z** überprüft, die bei Filmstart mit dem Wert 1 initialisiert wurde.

```
on (release) {
   if (z<3) {
      gotoAndStop ("eingabe");
      z++;
   } else {
      gotoAndStop ("end");
   }
}
```

Hier könnte man nun z. B. auch ein Cookie setzen, das überprüft wird und es dem User erschwert, bei einem Neustart der Seite wieder drei Versuche zu haben.

Achtung: Das Textfile mit den Daten muss natürlich an einem sicheren Platz auf dem Server liegen. Es darf nicht durch eine direkte Eingabe lesbar sein, sondern sollte sich in einem geschützten Bereich befinden. Eine andere Möglichkeit ist es, die Daten aus einer Datenbank zu beziehen oder eine Verschlüsselung einzubauen.

Verschlüsselungstechnik in Flash

▶ Roland Grela, www.flash-your-life.de

Wie gerade thematisiert, gibt es ein Problem bei einfachen Passwortabfragen wie zum Beispiel:

```
if (Eingabe eq "geheimes Passwort") {
   getURL ("geheim.htm");
}
```

Das ist kein Hindernis für Hacker. Man muss sich nur die SWF-Datei herunterladen (bzw. im Cache suchen), mit einem Texteditor oder Hex-Editor öffnen, und schon hat man das Passwort und die Datei, auf die man gelangt, wenn man das richtige Passwort eingegeben hat.

Man kann seinen Passwortschutz an vielen Stellen verbessern und somit das Erlesen erschweren oder sogar (fast) unmöglich machen. Als Erstes sollte man neben dem Passwort auch noch einen Benutzernamen abfragen. Somit hat man später die Möglichkeit, bestimmte Personen zu sperren oder jeden auf eine individuelle Datei umzuleiten.

Das Wichtigste ist allerdings die Verschlüsselung des Passwortes! Wenn man mit lauter Variablen herumspielt und aus diesen Passwort, Benutzername und URL errechnen lässt, verliert der Leser der SWF-Datei den Durchblick.

Wie es nicht mehr geht!

Man sollte annehmen, dass ein Schutz, der etwa so aussieht, recht sicher wäre.

```
if (user eq (4 * 3 - 5) and pass eq (37 * 3 - 5)) {
   getURL ((23 * 17 + 141) add (5 * 75) add ".htm"));
}
```

In Flash 4 ging das problemlos. Die beiden Eingabefelder heißen in diesem Beispiel `user` und `pass`, der Benutzername wäre »7« gewesen, das Passwort 106 und die URL `532375.htm`, und nichts von dem hätte man in der SWF-Datei erkennen können.

Doch leider exportiert Flash 5 eine SWF-Datei anders als Flash 4. Flash in der fünften Version führt diese Rechnungen gleich beim Export aus, und so werden die Daten wieder wie im ersten Beispiel im Klartext gespeichert.

Wie es richtig geht!

Man kann verhindern, dass Flash 5 die Ergebnisse von Berechnungen in der SWF-Datei speichert, indem man Berechnungen mit Variablen durchführt.

```
on (release) {
   a = "5";
   b = "4";
   c = "12";
   d = "3";
```

```
    if (user eq (a * b) and pass eq (Number(c) + Number(d))) {
    getURL ((a * c) add (a * b + Number(d)) add ".htm"));
    }
}
```

Am Anfang des ActionScripts legen wir die Werte der Variablen fest. Diese erscheinen dann natürlich auch wieder in der SWF-Datei. Erst `on (release)`, also wenn man auf den Button klickt, in dem dieses Skript steht, setzt Flash die Werte der Variablen in die Rechnung ein. Wenn dann also der Benutzername »20« und das Passwort »15« ist, wird die Datei 6023.htm aufgerufen. Umsteiger, die von Flash 4 gewohnt sind, `(c + d)` zu schreiben, müssen darauf achten, dass Flash 5 in diesem Fall die beiden Werte logisch verknüpft und somit das Passwort »123« wäre! Der Dateiname wird dann aus den Variablen am Anfang errechnet. Die Zwischenergebnisse sind »60« aus dem ersten Teil und »23« aus dem zweiten. Diese werden dann verknüpft und bekommen noch ».htm« angehängt. Unsere Datei heißt dann also 6023.htm.

Der Algorithmus

Da in dem obigen Beispiel die Variablen aber in der SWF-Datei zu lesen sind und unsere Rechnung nicht gerade komplex ist, kann ein Hacker sehr schnell an das Passwort kommen, indem er einfach alle möglichen Berechnungen durchgeht.

Wichtig ist also, dass wir einen komplexeren Algorithmus verwenden, der nicht so leicht zu erraten ist. Ihrer Kreativität sind dabei keine Grenzen gesetzt, vorrausgesetzt Flash 5 unterstützt Ihre Rechnung.

Wörter als Benutzername, Passwort oder Dateiname

Da es nun einmal Benutzer**name** und Pass**wort** heißt, wollen Sie wahrscheinlich auch in den Genuss kommen, Buchstaben zu verwenden, vor allem weil diese oft einfacher zu merken sind als Zahlenkombinationen. Um dieses zu ermöglichen, müssen wir die Stringfunktion **»chr«** verwenden, die den ASCII-Wert (American Standart Code of Information Interchange) in einen Buchstaben (oder in ein anderes Zeichen) konvertiert.

Eine ASCII-Tabelle finden Sie im Anhang auf Seite 670.

Wir nehmen also jeden Buchstaben unserer gewünschten Daten, Benutzername, Passwort und Dateiname, und suchen uns davon den ASCII-

Wert heraus. Allerdings dürfen wir in unserem Skript nicht einfach "chr(Zahl) add chr(Zahl)...." schreiben, da Flash auch dieses direkt konvertieren würde und die Daten am Ende wieder klar lesbar sind. Deswegen müssen wir den umständlichen Weg gehen:

```
on (release) {
   a = "116";
   b = "101";
   c = "115";
   d = "117";
   e = "114";
   f = "108";
   if (user eq (chr(a) add chr(b) add chr(c) add chr(a)) and pass eq
(chr(a) add chr(b) add chr(c) add chr(a))) {
      getURL (chr(d) add chr(e) add chr(f) add ".htm");
   }
}
```

Am Anfang dieses Skripts bekommen die Variabeln die ASCII-Werte der Buchstaben, die wir verwenden. In diesem Fall also t, e, s, u, r und l. In der IF-Schleife werden die Werte dann in die entsprechenden Buchstaben konvertiert und logisch verknüpft. Benutzername und Passwort sind "test", und wenn die Eingabe richtig war, wird man zu "url.htm" geschickt. Theoretisch könnten wir auch "ord(Buchstabe)" beim Zuweisen der Variablen verwenden und müssten nicht selbst nachsehen, welchen ASCII-Wert die Buchstaben haben; doch da wir gleich dieses Skript noch verändern, habe ich auf "ord" verzichtet. Zudem gibt es auch ein Programm, das die Werte der Variablen einer SWF-Datei auslesen kann.

Wie Sie sich wahrscheinlich denken können, sind die ASCII-Werte in der SWF-Datei lesbar. Das ist natürlich wieder ein Sicherheitsproblem, da man als Hacker herausfinden kann, dass es sich hierbei um ASCII-Werte handelt und wie man sie zum passenden Benutzernamen oder Passwort verknüpft.

Aus diesem Grund sollten wir wieder einen Algorithmus verwenden. Unser ActionScript könnte dann so aussehen:

```
on (release) {
   a = "57";
   b = "50";
   c = "58";
```

```
   d = "56";
   e = "53";
   if (user eq (chr(a * 2 + 2) add chr(b * 2 + 1) add chr(a * 2 + 1)
add chr(a * 2 + 2)) and pass eq (chr(a * 2 + 2) add chr(b * 2 + 1) add
chr(a * 2 + 1) add chr(a * 2 + 2))) {
     getURL (chr(c * 2 + 1) add chr(d * 2 + 2) add chr(e * 2 + 2) add
".htm");
   }
}
```

Unser Algorithmus ist in diesem Beispiel 2 + 2 bzw. 2 + 1, je nachdem, ob der ASCII-Wert gerade oder ungerade ist. Die Werte der Variablen müssen wir dann natürlich auch entsprechend ändern: den ASCII-Wert durch 2 dividieren und danach, wenn der ursprüngliche Wert gerade war, 2 subtrahieren oder, wenn der ursprüngliche Wert ungerade war, 1 subtrahieren. Sie sollten den Variablen nicht den Wert »ASCII-Wert / 2 – 2« bzw. »ASCII-Wert / 2 – 1« geben, da dann der Algorithmus für einen Hacker offensichtlich wäre!
Natürlich gilt auch hierbei wieder: Ihrer Kreativität, was den Algorithmus betrifft, sind (fast) keine Grenzen gesetzt.

Bitweise Verschlüsselung

Ein weitere sichere Methode der Verschlüsselung in Flash 5 ist die bitweise Verschlüsselung. Dazu muss man allerdings wissen, wie man Dezimalzahlen in binäre umrechnet.

Im Dezimalsystem gibt es die Ziffern 0, 1, 2, 3, 4, 5, 6, 7, 8 und 9. Das Binärsystem besteht hingegen nur aus zwei Ziffern: 0 und 1. Man rechnet vom Dezimal- ins Binärsystem, indem man die Dezimalzahl durch 2 dividiert und den Rest (1) oder den Nichtrest (0) notiert. Das Ergebnis dividiert man dann wiederum durch 2 und schreibt den Rest oder Nichtrest unter den vorherigen. Diese Rechnung führt man bis zum Ende durch.

Ein Beispiel mit der Dezimalzahl 23:

```
23/2 = 11 R 1
11/2 =  5 R 1
 5/2 =  2 R 1
 2/2 =  1 R 0
 1/2 =  0 R 1
```

Man liest dann die Reste von unten nach oben (!) und erhält so aus der Dezimalzahl 23 die Binärzahl 10111.

Um jetzt eine Binärzahl im Dezimalsystem darzustellen, muss man folgende Rechnung durchführen:

b1*2^0 + b2*2^1 + b3*2^2 + b4*2^3 + ...

b1, b2, b3, b4, ... entsprechen den einzelnen Stellen der Binärzahl von rechts nach links (!). Unsere Binärzahl 10111 wird dann folgendermaßen in die Rechnung eingesetzt:

1011 B = 1*1 + 1*2 + 1*4 + 0*8 + 1*16 = 23 Dezimal

Beispiele der bitweisen Verschlüsselung

Um jetzt mit binären Zahlen zu rechnen, benötigen wir die bitweisen Operanden, die bei Flash 5 neu dazugekommen sind. Zu den bitweisen Operanden gibt es eine gute Erläuterung in den Flash 5-Hilfedateien im ActionScript-Wörterbuch.

```
on (release) {
   a = "2";
   b = "3";
   c = "31";
   if (user eq ((Number(a)+Number(b))<<b) and pass eq
((Number(c)+Number(b))>>a)) {
      getURL (((c-a)<<(c-1)) add ".htm");
   }
}
```

Dieses ActionScript sieht auf den ersten Blick wohl sehr kompliziert aus, aber das soll es ja auch.

Wenn der Benutzername dem Ergebnis von »5<<3« entspricht und das Passwort dem von »34>>2«, wird die HTM-Datei aufgerufen, deren Name dem Ergebnis von »29<<30« entspricht.

5 entspricht der Binärzahl 101. Diese wird nun um drei Stellen nach links verschoben, und die leeren Stellen werden mit 0 gefüllt. Der Benutzername entspricht also der Binärzahl 101000, die im Dezimalsystem 40 ist.

34 entspricht der Binärzahl 100010. Diese wird um zwei Stellen nach rechts verschoben, wobei die letzten beiden Ziffern wegfallen. Das Passwort entspricht demnach der Binärzahl 1000, die im Dezimalsystem 8 ist.

Wenn man folglich »40« als Benutzername und »8« als Passwort eingibt, wird eine URL geöffnet, die den Namen 1073741824.htm trägt. Die Dezimalzahl 29 ist binär ausgedrückt 11101. Da Flash 5 32-Bit-Schlüssel verwendet, kann eine Binärzahl auch nur 32 Stellen haben. Deshalb beträgt 11101, nachdem diese Zahl um 30 Stellen nach links verschoben wurde, 10000000000000000000000000000000 (31 mal 0), da hier die ersten drei Stellen, die aus der Zahl eine 34-Stellige machen, wegfallen. Diese Binärzahl entspricht dann der Dezimalzahl 1073741824 (da 1*2^30 = 1073741824). Indem der String ».htm« angehängt wird, heißt die Datei 1073741824.htm.

Als Letztes will ich noch ein langes Beispiel bringen, das mit Binärzahlen rechnet, Wörter als Daten und auch mehrere Benutzer mit verschiedenen Passwörtern erlaubt.

```
on (release) {
   a = "5";
   b = "23";
   c = "1095";
   // verschlüsselter Benutzername und Passwort von Person 1
   if (user eq ((chr((c-560)/a-b)) add (chr((c-425)/a-b)) add
(chr((c-490)/a-b)) add (chr((c-455)/a-b)) add (chr((c-495)/a-b))
add (chr((c-405)/a-b))) and pass eq ((chr((c-645)/a-b)) add
(chr((c-410)/a-b)) add (chr((c-495)/a-b)) add (chr((c-370)/a-b))
add (chr((c-375)/a-b)) add (chr((c-560)/a-b)) add (chr((c-425)
/a-b)) add (chr ((c-490)/a-b)))) {
      getURL (((b-a)<<15)*c add ".htm");
   // verschlüsselter Benutzername und Passwort von Person 2
   } else if (user eq ((chr((c-600)/a-b)) add (chr((c-495)/a-b))
add (chr((c-410)/a-b)) add (chr((c-405)/a-b))) and pass eq
((chr((c-595)/a-b)) add (chr((c-425)/a-b)) add (chr((c-475)/a-b))
add (chr((c-420)/a-b)) add (chr((c-420)/a-b)) add (chr((c-375)
/a-b)))) {
      getURL (((b-a)<<15)*c add ".htm");
   // verschlüsselter Benutzername und Passwort von Person 3
   } else if (user eq ((chr((c-595)/a-b)) add (chr((c-495)/a-b))
add (chr((c-400)/a-b)) add (chr((c-400)/a-b)) add (chr((c-460)/
a-b)) add (chr((c-455)/a-b)) add (chr((c-495)/a-b)) add
(chr((c-405)/a-b))) and pass eq ((chr((c-565)/a-b)) add
(chr((c-420)/a-b)) add (chr((c-475)/a-b)) add (chr((c-495)/a-b))
add (chr((c-410)/a-b)) add (chr((c-435)/a-b)) add (chr((c-455)
```

```
/a-b)) add (chr((c-430)/a-b)) add (chr((c-400)/a-b)))) {
      getURL (((b-a)<<15)*c add ".htm");
   } else {
      getURL ("falsch.htm");
    }
}
```

Ich nehme einmal an, dass Sie sich nicht die Mühe gemacht haben, Benutzernamen, Kennwörter und Dateiname zu errechnen. Deswegen liste ich sie Ihnen noch einmal auf:

```
Benutzer1 = Tobias
Passwort1 = CrazyTob
Benutzer2 = Lars
Passwort2 = Moeppy
Benutzer3 = Matthias
Passwort3 = Spearmint
Dateiname = 645857280.htm
```

Der verwendete Algorithmus von Benutzernamen und Passwörtern ist `(chr((c-x)/a-b)) add ...`. Für `x` habe ich bei jedem Buchstaben eine andere Zahl eingesetzt. Diese wird errechnet, indem man zu dem ASCII-Wert des Buchstabens 23 addiert und dann mit 5 multipliziert. Dieses Ergebnis muss dann von 1095 subtrahiert werden. Als Skript wird es so aussehen: »1095-(ord(Buchstabe)+Number(5))*23«

Die URL wird wieder durch binäre Verschlüsselung errechnet, indem man die Dezimalzahl 19 in eine Binärzahl konvertiert und dann um 15 Stellen nach links verschiebt.

Passwortabfrage mit Flash und CGI

◀ Marco Asbach www.partylogger.de

Noch eine Möglichkeit ist, die Passwortabfrage über ein Skript zu regeln. Dabei sind viele weitere Features möglich. Außerdem liegt ein Skript in der Regel an einem sicheren Platz des Servers und erschwert so die Dekodierung.

Folgende Features bietet diese Variante:

- Beim ersten Aufruf des Filmes wird über die Flash-Benutzeroberfläche das Passwort festgelegt.
- Das Passwort wird verschlüsselt in eine *.dat-Datei auf dem Server gespeichert.

- Bei dem Versuch, sich mit einem falschen Passwort Zugang zu verschaffen, wird der Webmaster hierüber per Mail informiert.
- In der Mail sind Uhrzeit, falsches Passwort und IP-Adresse desjenigen angegeben, der dies versucht hat.

Auf der CD erhalten Sie folgende Dateien:
passwort.fla => der Flash-Film mit der Passwortabfrage
schutz.pl => das dazugehörige CGI-Skript (geschrieben in Perl)
passwort.dat => die Containerdatei für das Passwort

Wie funktioniert die Passwortabfrage?

Die komplette Funktion ist in einem MovieClip integriert. Bei Aufruf des Filmes wird im ersten Frame des MovieClips folgendes ActionScript ausgeführt:

```
status = "....bitte warten...." add chr(13) add "Passwort wird
initialisiert";
action = "proof";
loadVariables ("cgi-bin/schutz.pl", "_root.abfrage", "POST");
```

Ein Hinweis wird mit der Variable `status` definiert, die in einem Textfeld angezeigt wird. Die Aktion `action` wird auf `"proof"` gesetzt, und mit dem Befehl `loadVariables` wird das CGI-Script aufgerufen, und die Variablen werden mittels Post zwischen dem Skript und dem MovieClip ausgetauscht. Diese Aktion `"proof"` veranlasst das CGI-Script zu prüfen, ob in der Datei passwort.dat schon ein Passwort enthalten ist. Je nachdem, wie das Ergebnis aussieht, wird an den Flash-Film die Variable `firsttime` mit `yes` oder `no` zurückgegeben.

Im zweiten Frame ist folgendes ActionScript:

```
if (ende == 1) {
 if (firsttime == "yes") {
  gotoAndStop ("firsttime");
 } else {
  gotoAndStop (4);
 }
}
```

Die Variable ende werden Sie noch öfter sehen, diese wird vom Skript nach Beendigung seiner Aktion an den Flash-Film zurückgegeben, damit dieser »weiß«, dass das CGI-Script seine Arbeit erledigt hat. Ist dies der Fall (ende == 1), wird die Variable "firsttime" überprüft. Ist der Inhalt "yes", wird der Film zum Bild mit der Bezeichnung "firsttime" verwiesen. Ist dies nicht der Fall (else), wird der Film zu Bild Nummer 4 verwiesen. Ist ende ungleich 1, dann wird dieses Bild einfach durchlaufen, und der Film kommt zu Bild Nummer 3, in dem folgendes ActionScript steht:

```
gotoAndPlay (2);
```

Damit wird der Film zurück auf Bild Nummer 1 verwiesen, und die Abfrage, ob "ende ==1" ist, kann erneut erfolgen. Ist bereits ein Passwort definiert worden, wird der Film zu Bild Nummer 4 verwiesen und stoppt an dieser Stelle, da der Verweis mit gotoAndStop erfolgte. In Bild Nummer 4 steht folgendes ActionScript:

```
ende = "0";
status = "Bitte geben Sie das Passwort ein!";
```

Hier wird die Variable ende auf "0" zurückgesetzt und die Statusmeldung neu definiert. In diesem Bild befindet sich ein Textfeld des Typs »Texteingabe« und der Variable »password«. Hier erfolgt im veröffentlichten Film die Eingabe des Passwortes. Unterhalb des Eingabefeldes sind zwei Schaltflächen (Buttons) mit der Bezeichnung »Löschen« und »Senden«. Mit dem Button Löschen wird eine vorgenommene Eintragung in das Textfeld wieder gelöscht:

```
on (release) {
   password = "";
}
```

Bei Auslösen des Buttons Senden wird folgendes ActionScript ausgelöst:

```
on (release) {
 action = "verify";
 loadVariables ("cgi-bin/schutz.pl","_root.abfrage","POST");
status = "....Passwort wird überprüft....";
play ();
}
```

Auch hier wird nun wieder, wie oben angegeben, das CGI-Script aufgerufen. Die Variable `action` mit dem Inhalt `verify` veranlasst das CGI-Script zur Überprüfung des im Textfeld eingegebenen Passwortes mit dem auf dem Server befindlichen verschlüsselten Passwort. Die Anweisung `play()` lässt den Movie-Clip weiterlaufen, so dass dieser in Bild Nummer 5 gelangt, das folgendes ActionScript enthält:

```
if (ende == 1) {
 gotoAndPlay (7);
}
```

Auch hier erfolgt über die Variable `ende` wieder die Prüfung, ob das CGI-Script seine Arbeit erledigt hat. Ist dies nicht der Fall (ende ungleich 1), gelangt der Film in Bild Nummer 6, das wieder mit einem `gotoAndPlay` auf Bild Nummer 5 verweist. Ist das Skript fertig (ende = 1), dann wird der Film zu Bild Nummer 7 verwiesen, das folgendes ActionScript beinhaltet:

```
if (correct == "yes") {
 _root.gotoAndPlay("Start");
 stop ();
}
```

Hier erfolgt die Überprüfung einer Variablen mit der Bezeichnung `correct`. Diese wird vom CGI-Script an den Flash-Film übermittelt. Sie kann den Wert `"yes"` oder `"no"` haben. Bei `"yes"` war das Passwort korrekt, und die Hauptzeitleiste `"_root"` wird angewiesen, zum Bild mit der Bezeichnung »Start« zu gehen und den Film fortzufahren. Ab diesem Bild in der Hauptzeitleiste können dann Ihre geschützten Teile des Films integriert werden. Ist der Inhalt `"no"`, dann war das Passwort falsch, und der MovieClip läuft weiter zu Bild Nummer 8, das auch die Bezeichnung »falsch« hat. Natürlich gibt es in diesem Bild auch wieder ein wenig ActionScript:

```
status = "Das von Ihnen eingegebene Passwort ist nicht korrekt!";
password = "";
fehler = "Das Passwort ist falsch! Der Inhaber der Domain wurde per
Mail darüber benachrichtigt. Folgende Informationen wurden versen-
det:" add chr(13) add "Zeit: " add zeit add chr(13) add
"Ihre IP: " add ip add chr(13) add "Versuch: " add wrong;
stop ();
```

Hier wird eine Warnmeldung definiert, die einige Angaben enthält, die vom CGI-Script übermittelt wurden. Es wird die Zeit (`"zeit"`), das falsche Passwort (`wrong`) und die IP-Adresse (`"ip"`) des Besuchers Ihrer Seite ausgegeben, der das falsche Passwort eingegeben hat. (Dient der kleinen Abschreckung, das CGI-Script leitet diese Daten auch per Mail an den Webmaster, sprich Sie, per Mail weiter.) Der Film wird mit `"stop()"` gestoppt. In diesem Bild gibt es einen ZURÜCK-Button, der wieder auf Bild Nummer 4 zur erneuten Passworteingabe verweist.

Solange also kein richtiges Passwort eingegeben wurde, kann der Film nicht abgespielt werden.

Wie passe ich diese Passwortabfrage an meine Bedürfnisse an?

Als Erstes ist es wichtig, das CGI-Script anzupassen. Öffnen Sie dazu das Skript (schutz.pl) in einem einfachen Texteditor, z. B. Notepad von Windows. In der ersten Zeile finden Sie Folgendes:

```
#!/usr/bin/perl
```

Dies ist die Pfadangabe zum Perl-Interpreter auf Ihrem Server (Linux-/Unix-Betriebssystem). Meist ist dieser unter diesem Pfad zu finden, einige Server sind anders konfiguriert. Wie Ihr Pfad lautet, erfahren Sie meist in den FAQs Ihres Providers oder bei Ihrem Systemadministrators. Wenn Sie diesen angepasst haben, finden Sie unterhalb meines Copyright-Hinweises folgende Zeilen:

```
# Hier steht der Pfad zu Ihrem Sendmail-Programm des Servers
$mailprog="/usr/lib/sendmail -t";

# Hier tragen Sie Ihre E-Mail-Adresse ein
# Vergessen Sie nicht den \ vor dem @
$yourmail="Ihre-Email\@provider.de";
```

Passen Sie den Pfad Ihres Sendmail-Programms an und tragen Ihre E-Mail-Adresse ein, denken Sie an den Backslash!! Den Pfad zum Sendmail gibt es auch wieder in den FAQs des Providers oder beim Systemadministrators.

Wenn Sie sich ein wenig in Perl auskennen, können Sie sicher noch andere kleinere Änderungen vornehmen, wenn Sie allerdings nicht Perlkonform sind, rate ich von weiteren Änderungen ab.

Nun zum *.fla, ich habe die Passwortabfrage bewusst in einen MovieClip gepackt, damit haben Sie die Möglichkeit, diesen per Copy and Paste in Ihre bestehenden Projekte einzubinden. Wichtig ist, dass dieser MovieClip möglichst im ersten Frame entweder vor oder nach einem Preloader eingesetzt wird. Was passiert, wenn das Passwort korrekt war, können Sie unter Umständen anpassen, indem Sie anstatt der Aktion:

```
_root.gotoAndPlay("Start");
```

in Bild Nummer 7 eine andere Aktion einsetzen. Nehmen Sie keine Änderung vor, müssen Sie in der Hauptzeitleiste das erste Bild mit geschütztem Inhalt noch mit der Bezeichnung »Start« versehen. (Ein Hinweis an dieser Stelle: Vielleicht kommen Sie nun auf die Idee, bei einem richtigen Passwort die Aktion in Bild Nummer 7 dahingehend zu ändern, dass Sie einen »Get URL« einbauen. Ich rate davon ab, da jeder veröffentlichte Film (*.swf) mittels einem Hex-Editor geöffnet und so die URL ausgelesen werden kann, von Schutz kann man dann sicher nicht mehr reden. Daher auch diese Möglichkeit der Passwortabfrage, denn jede **nur** in Flash realisierte Passwortabfrage ist nicht sicher.)

Sie können natürlich das Design dem Ihren anpassen und auch die Statusmeldungen verändern, seien Sie dabei aber sorgfältig, damit nicht der wichtige ActionScript-Teil verändert wird.

Alle im ActionScript angegebenen Pfade beim Aufruf des CGI-Scripts sind daraufhin ausgelegt, dass sich das Skript in einem Verzeichnis cgi-bin befindet und der Film eine Verzeichnisebene höher liegt. Sollte Ihre Serverarchitektur eine andere sein, müssen Sie gegebenenfalls alle LoadVariables anpassen.

Wie lade ich die angepassten Dateien auf meinen Server?

Ein gutes FTP-Tool ist WS_FTP. Laden Sie den veröffentlichten Film (*.swf) im Binary-Modus auf Ihren Server. Die Datei schutz.pl und passwort.dat laden Sie bitte im ASCII-Modus auf den Server (ganz wichtig, dass dies so geschieht, da dass Skript ansonsten nicht funktioniert). Nun müssen Sie der Datei schutz.pl und passwort.dat noch Rechte zuweisen, den so genannten `chmod`. Für das Skript muss dieser 755 und für die *.dat-Datei 666 sein.

Wenn dies erfolgt ist, können Sie Ihren Film das erste Mal aufrufen. Wenn alles korrekt ist, sollte Sie der Film nun zur Eingabe und Festlegung Ihres Passwortes auffordern. Dieses Passwort muss zur Sicherheit zwei Mal eingegeben werden. Mit dem Button SENDEN wird dieses dann verschlüsselt in der *.dat-Datei abgespeichert, und der Film kehrt zur regulären Passwortabfrage zurück.

Voilà, fertig ist Ihr passwortgeschützter Film. Sollte es nicht klappen, lesen Sie bitte noch den Abschnitt »Troubleshooting«.

Troubleshooting

Anbei eine Liste, die sich aus den FAQs der Nutzer meiner Quelldateien ergeben hat:

1. Datei-Upload
   Der Upload von CGI-Scripts und der dazugehörigen *.txt- oder *.dat-Dateien muss immer im ASCII-Modus erfolgen, ansonsten kann der Perl-Interpreter Ihres Servers damit nichts anfangen.
2. Das CGI-Script
   Es gibt Server, z.B. Puretec, die keine CGIs mit der Endung *.pl mögen, dort werden nur Dateien beachtet, welche die Dateiendung *.pl (für Perl-Script) haben; das CGI dann einfach umbenennen. Dies kann auch umgekehrt der Fall sein. Ferner muss im CGI-Script selbst der Pfad zum Perl-Interpreter angegeben werden. Meist ist dies #!/usr/bin/perl, aber er kann auch anders lauten. Um dies genau zu erfahren, gehen Sie auf die FAQs Ihres Providers. Die Antwort werden Sie in den meisten Fällen dort finden, ansonsten schreiben Sie den Admin Ihres Servers an. Noch ein Zusatzhinweis: Speziell Linux-/Unix-Server sind sehr empfindlich, was die Groß-/Kleinschreibung angeht, also auf jeden Fall strikt darauf achten, denn die Datei schutz.pl ist eine andere als Schutz.pl.
3. Rechte
   Werden Ihre Dateien auf einen Linux-/Unix-Server aufgespielt, ist es erforderlich, den Dateien Zugriffsrechte zu geben, den so genannten chmod. Dieser muss für die *.cgi oder *.pl immer 755 und für die *.txt oder *.dat 666 lauten. Weitere Infos unter: http://internet-partner.de/webmaster/cgi_service/CGIManual.html .
4. Zum *.fla
   Modifizierungen am *.fla sind mit Vorsicht vorzunehmen. Packen Sie z.B. den MovieClip in einen anderen MovieClip, stimmen die Variablen nicht mehr, und das Ganze funktioniert nicht.

5. Sonstiges
   Bei manchen Providern spielen Sie zwar Ihre Datei per FTP ins Verzeichnis namens cgi-bin, allerdings erfolgt über das Web der Aufruf über eine andere Verzeichnisbezeichnung, z. B. cgi-domainname.
6. Online-Test
   Sie können die CGI-Scripts auch einfach online testen, indem Sie diese direkt über den Browser ansprechen (http://www.domainname.de/cgi-bin/name.cgi). Wenn Sie vom Server einen HTTP 500-Fehler bekommen, ist das Skript nicht richtig konfiguriert oder im falschen Modus aufgespielt worden. Bekommen Sie einen HTTP 404-Fehler, ist der Link falsch bzw. der Verzeichnisname doch ein anderer. Bei einem funktionierenden Passwort-Script bekommen Sie die Rückmeldung »firsttime=irgendetwas&ende=1«, wobei irgendetwas für »yes« oder »no« steht.
7. Worst Case
   Tja, und leider läuft das Skript bei einigen wenigen Servern überhaupt nicht, dies kann z. B. daran liegen, dass eine zu alte Perl-Version aufgespielt ist. Voraussetzung für dieses Skript ist eine Perl-Version ab 5.

Flash 5 Detection

Jedem das Seine

Vor etwa einem Jahr haben wir aus der Not, dass keine zufriedenstellende Flashdetection existierte, gut zwei Monate mit Testen und Experimentieren verbracht, um eine funktionierende Flashdetection zu erfinden. Heute, mit der fünften Version von Flash, ist sie wichtiger denn je.

▸ Carlo Blatz

Seit der dritten Version von Flash ist es für Flash-Entwickler notwendig geworden, die installierte Version des Flash-Plug-ins zu überprüfen. Dank dem Softupdate kann bei neueren Browsern auch ohne Detection immerhin überprüft werden, ob Flash installiert ist oder nicht. Wenn wir aber wissen möchten, ob Flash 2, Flash 3, Flash 4, Flash 4 mit Printfeature, Flash 4 mit domainübergreifenden CGIs oder Flash 5 installiert ist, muss man sich schon etwas einfallen lassen. Mit Flash 5 kreierte HTML-Files können nun endlich das Softupdate von Flash 5 einleiten, auch wenn Flash 4 bereits installiert ist. Spätestens wenn wir das aber verhindern wollen, Flash 4-kompatible Files publizieren oder sogar für mehrere Flash-Versionen ein File anbieten möchten, benötigen wir eine Flashdetection.

Flashdetections gibt es schon seit einigen Jahren. Leider funktionieren die meisten nicht so wie gewünscht. Macromedia selbst bietet eine Flashdetection an, die auf JavaScript basiert – was aber, wenn JavaScript deaktiviert ist? Moock hat die erste Flash-basierte Flashdetection kreiert, leider funktionierte sie nicht auf dem Macintosh. Sie hatte auch einen Fehler – um Flash 4 zu erkennen, wurde eine Variable gesetzt, und wenn diese Variable erkannt wurde, sollte zur Flash 4-Seite umgeleitet werden. Leider hat Flash 3 einfach die IF-Abfrage ignoriert (weil Flash 3 das noch nicht konnte), die Bedingung aber trotzdem ausgeführt – denn GetURL kannte Flash 3 bereits.

Einem Kunden kann man nicht verkaufen, dass man nicht sicher zwischen Flash 3 und Flash 4 unterscheiden kann, und schon gar nicht, dass die ganze Detection nicht auf dem Macintosh funktioniert. Also haben

wir einige Wochen experimentiert und mit den Entwicklern der »Problemkinder« Kontakt aufgenommen. Gemeinsam haben wir eine Detection entwickelt, die bis Flash 4 nahezu perfekt zwischen NoFlash, Flash 2, Flash 3 und Flash 4 unterscheiden konnte. Bis auf eine Version des Internet Explorers 4.71 (die ein grundsätzliches Problem mit der Plug-in-Initialisierung hatte) haben wir keine Ausfälle gefunden. Mit dem Printfeature und dem neuen Plug-in, das keine domainübergreifenden Skripte mehr unterstützte, wurde es aber komplizierter. Wir konnten nicht mehr mit versionsbedingten Aktionen überprüfen, sondern mussten auf die Flash 4-interne Variable `$version` zurückgreifen. Diese Variable wird aber nicht vom Unix-/Linux-Plug-in unterstützt. Es gibt also einen Ausfall von durchschnittlich 5 %.

Zur Theorie

Ein Flashfile wird in einem HTML-File aufgerufen, das nach zehn Sekunden automatisch zur NoFlash-Seite weiterleitet (MetaRefresh). Im Flashfile wird versucht, vor diesem zehn Sekunden-Countdown zur entsprechenden Flash-Seite umzuleiten. Dazu fängt man hinten an und versucht erst Flash 5 zu erkennen, dann Flash 4, Flash 3 und Flash 2. Diese Detection funktionierte aber nun auf dem Microsoft-eigenen Windows NT-System nicht, und wenn sie im Internet Explorer funktionierten, dann doch meist in Netscape nicht. Damit diese Detection möglichst unsichtbar verläuft, wurden die meisten Flashdetections auf der minimalen Pixelgröße 18 x 18 Pixel erstellt. Einige haben das Flashfile sogar noch mit vielen Zeilenumbrüchen aus der HTML-Seite gerückt, aber eben da liegt das Problem. Die Detection muss sichtbar sein, damit sie rechtzeitig initialisiert wird, und dafür muss sie auch deutlich größer als 18 Pixel sein. Es geht sogar noch weiter, es muss etwas dargestellt und gestreamed werden, damit z. B. Macintosh Netscape rechtzeitig das Plug-in initialisiert. Wir schalten also als Allererstes ein Formtweening. Damit die Detection unsichtbar ist, ist die Hintergrundfarbe des SWFs, des HTMLs und die Farbe des Formtweenings die gleiche websichere Farbe.

Kommen wir zum Skript

Es ist so angelegt, dass es auch mit Flash 4 erstellt werden kann. Die Detection ist zur Sicherheit 200 x 200 Pixel groß, spielt zwölf Bilder pro Sekunde und beginnt mit einem Formtweening in drei Keyframes von ei-

nem Kreis zu einem Viereck. Die Detection fängt hinten an. Sie versucht zuerst Flash 5 zu erkennen, indem die erste Zahl der Eigenschaft `$version` ausgelesen und in die Variable `Version` geschrieben wird.

```
version = substring(eval("$version"), 5, 1);
```

Wenn diese Zahl nun 5 entspricht, wird die Aktion des Frames »Flash 5« aufgerufen. Die Aktion »Call« gibt es erst seit Flash 4, kommt also ein Flash 2- oder 3-Player, kennt er weder die Bedingung If noch die Aktion `Call` und findet das `play`. Im Frame »Flash5« (das am Ende der Zeitleiste steht) wird nur ein GetURL der Flash 5-Seite aufgerufen und natürlich ein Stopp gesetzt.

```
if (version ge "5") {
    call ("Flash5");
} else {
    play ();
}
```

Wenn `Version` also nicht 5 entspricht, wird mit einigen Frames Abstand (falls ein Rechner zu schnell ist und die Aktion des nächsten Frames ausführt, bevor das GetURL von Flash 5 ausgeführt ist) die Flash 4-Print-Abfrage gestartet. Das Flash 4-Plug-in mit Printfeature war das erste Plug-in mit der Eigenschaft `"$version"`, es reicht also zu überprüfen, ob es diese Eigenschaft gibt.

```
if (eval("$version") gt "") {
    call ("Flash4Print");
} else {
    play ();
}
```

Wenn also weder Flash 5 noch Flash 4 Print gefunden wurde, nutzen wir einfach die Abwärtskompatibilität der Plug-Ins und starten wiederum einige Frames später einen einfachen CALL des Frames »Flash4«. Dort ist auch nur ein GetURL der Flash 4-Seite und ein Stopp.

```
call ("Flash4");
```

Ein Flash 3-Plug-in kennt diese Aktion nicht und führt einige Frames später ein GetURL in einem MovieClip aus. MovieClips gab es bei Flash 2 noch nicht. Aus dem MovieClip müssen wir nun aber auch die Hauptzeitleiste stoppen, damit dort nicht die Flash 2-Aktion einige Frames später auf der Hauptzeitleiste ausgeführt wird. Sicherheitshalber sollte man die Flashdetection noch aus Flash 4 exportieren.

Speeddetection

Mit immer neuen Ideen geht es auch immer mehr an die Grenzen der Prozessoren. Nur, wie schnell ist der Rechner des Users?

Seit jeher gibt es bei der Darstellung dynamischer Grafiken ein Problem: Die so genannte Drop-out-Rate. Die Drop-out-Rate stellt den Wert der Einzelbilder dar, die der PC nicht dargestellt »rausfallen« lässt.

Eine Animation mit 20 Einzelbildern die Sekunde ist an sich schon zu langsam, um wirklich flüssig zu erscheinen (mindestens 25 Frames per Sek. benötigt das menschliche Auge), und wenn da aus Performance-Gründen nun noch fünf Frames der Drop-out-Rate zum Opfer fallen, wird es wirklich unschön. Passiert das schon auf einem modernen Mittelklasse-PC, kann man sich ausmalen, mit wie vielen Einzelbildern ein User von einem 230er-PII versorgt wird.

Flash stellt, wenn kein Streamingsound läuft, alle Bilder dar, benötigt unter Umständen nur drei Sekunden statt eine für 20 Bilder. Das können wir uns zu Nutze machen.

Es bietet sich also an, optimierte Versionen allzu komplexer Animationen für User mit schmalbrüstigen Rechnern zu erstellen. Bei aufwändigen Effekten wie `DuplicateMovie` kann man eventuell auch die Anzahl der duplizierten MovieClips durch eine Variable je nach Rechner reduzieren oder erhöhen – nur wie erhält das richtige System die passende SWF?

Eine Möglichkeit wäre, dem User diese Auswahl zu überlassen – aber warum, wenn wir dies automatisieren können? Der folgende Absatz ist recht simpel. Auf der CD befindet sich ein genauer Framerat-Messer, mit dem man eine Weiterleitung anhand der exakten Framerate durchführen kann.

Eine aufwändige Animation mit möglichst vielen Linien wird ein bis drei Sekunden gedreht, gestaucht und ins Alpha geblendet – alles zur gleichen Zeit. Das sollte also etwa 40 bis 60 Frames dauern bei einer Framerate von 20 fps. Als Objekt bietet sich ein einfaches Gitter an – mit möglichst vielen Feldern.

Am ersten Frame Speeddetection (also nach erfolgreichem Preloading) starten wir die Funktion `GetTimer`. Die Funktion `GetTimer` gibt die vergangene Zeit vom Start des Films in eine Variable aus – nennen wir diese `time1`. Wir benötigen diese Variable, um nach der Animation zu sehen, wie viel Differenz zu dieser Zeit entstanden ist.

Der letzte Frame prüft nun die Drop-out-Rate. Wir ermitteln eine neue Variable `Time2` und ziehen von ihr `Time1` ab. Diesen Wert schreiben wir in die Variable `Time`. Ist der Wert der Variable `Time` in etwa drei Sekunden groß – bei einer 60 Frames langen Animation hat der PC keine oder nur ein bis zwei Frames ausgelassen –, kann er also aufwändige Objekte problemlos darstellen.

Es sollte noch ein Mittelklassewert und ein Wert für einen wirklich langsamen PC ermittelt werden. Ein PC, der für den 60 Frames langen Film vier Sekunden benötigt hat, muss also ganze 20 Bilder langsamer gewesen sein.

Hier noch mal das Skript zum Mitschreiben:

Keyframe 1 der Speeddetection (nach Preload):

```
TimeA = getTimer ();
```

```
Keyframe 60:
TimeB = getTimer ();
TimeX = TimeA - TimeB;
if (timeX <3200) {
    gotoAndStop ("schneller_PC");
} else if (timeX>=3200 and timeX<3400){
    gotoAndStop ("durchschnittlicher_PC");
} else if (timeX>=3400){
    gotoAndStop ("langsamer_PC");
}
stop();
```

Nach diesen Informationen können nun `Gotos` zu der entsprechend abgespeckten oder zu der Originalanimation gesetzt werden. Natürlich ist etwas Tuning nötig. Zudem sollte man das Ganze auf mindestens fünf PCs mit unterschiedlicher Hardwarekonfiguration testen.

Preloader

Wenn Sie größere Flash-Filme erstellen, können Sie mit einem Preloader das Laden des Films steuern. Wie dies funktioniert und ob das Preloaden immer sinnvoll ist, erfahren Sie in diesem Kapitel.

▸ Carlo Blatz

Man sollte es möglichst vermeiden, irgendwo einen Preloader zu verwenden – zumindest wenn das, was während des Preloadens zu sehen ist, den User langweilt. Flash-Seiten sind meist größer als HTML-Seiten, weil viel mit Animation, Sound und Pixelgrafiken gearbeitet wird. Warum auch nicht – dafür ist es ja da; nur sollten dann die Inhalte streamen. Niemand schaut sich gern einige Minuten »1 % – 2 % – 3 % ...« an.

Ein Ansatz wäre z. B., verschiedene Animationen zu loopen, so lange, bis ein bestimmter Teil des Films geladen ist. Wenn man das mit mehreren unterschiedlich langen Filmsequenzen macht, die gleichzeitig auf der Bühne liegen, ergibt sich ein zufälliges Muster von Animationen, das auch spannend sein kann. Solche zufälligen Animationen kann man natürlich auch mit ActionScript machen. Idealerweise hat man aber ein spannendes Intro, das genügend Zeit schindet, um den Rest der Seite zu laden.

Wenn es denn doch ein Prozentlader sein muss, gibt es mehrere Möglichkeiten:

Flash 2-kompatibler Preloader

Man erstellt die Animation eines größer werdenden Balkens über 100 Frames. Eine Ebene darunter kommt eine Fläche über die gesamte Bühne in der Farbe des Hintergrunds. Darunter eine Ebene mit 100 Keyframes, in die wir nun alle Symbole des fertigen Films verteilen. Diese Symbole sind nun nicht sichtbar, der Fortschritt der 100 Frames kann aber nur in der Geschwindigkeit des Streamings erfolgen. Nach 100 Frames ist dann alles geladen. Ein Nachteil dieser Variante ist

1. die etwas größere Datei durch mehr verwendete Instanzen und
2. die Unverhältnismäßigkeit zwischen Frames und Bytes.

Ein Symbol im ersten Frame kann 10 KB sein und ein anderes im 10. Frame nur 1 KB – der angezeigte Ladefortschritt entspricht also nur den geladenen Frames und nicht denen der geladenen Bytes.

Flash 3-kompatibler Preloader

Ab dieser Version haben wir die Aktion `If Frame is loaded`. In diesem Prinzip können wir die zu diesem Zeitpunkt geladenen Frames abfragen. In Worten: »Wenn Frame X geladen ist, dann mache das.« Wenn der Frame nicht geladen ist, wird die Aktion in der Bedingung ignoriert. Ein Frame später müsste also eine Schleife stehen, die Flash wieder in die Bedingung zurückspringen lässt, so dass andauernd abgefragt wird, ob dieser Frame geladen ist oder nicht. Möchte man so einen Prozentloader bauen, müsste man 100 Abfragen haben und 100 Schleifen, also mindestens 200 Frames.

Beispiel Movie mit 100 Frames:

Frame 1:

```
If Frame is loaded (Szene1, 10)
  Goto and play (3)
End if
```

Frame 2:

```
Goto and play (1)
```

Frame 3:

```
If Frame is loaded (Szene1, 20)
  Goto and play (5)
End if
```

Frame 4:

```
Goto and play (3)
```

Für einen Ladebalken eignet sich diese Variante nicht. Man könnte aber alle zwei Frames eine neue Prozentzahl zeigen. Auch hier haben wir wieder den Nachteil, dass die geladenen Frames nicht den geladenen Bytes

entsprechen. Mit großem Aufwand könnte man natürlich mittels des Bandbreiten-Profilers die Größe der einzelnen Frames ermitteln und so keine lineare Abfrage (in diesem Beispiel alle 10 Frames) erstellen, sondern tatsächlich die Frames errechnen, die einem Prozent der Datenmenge entsprechen.

Flash 4-kompatibler Preloader

Hier wird es dann gleich viel komfortabler. Wir haben zwei Variablen namens `_framesloaded` und `_framestotal`. Mit einfacher Prozentrechnung können wir so 1 % ermitteln und das mit den `_framesloaded` multiplizieren. Diesen Wert könnte man dann in einem Textfeld darstellen lassen. Erstellt man sich also ein Textfeld mit dem Variablennamen x und setzt in ein Keyframe die Aktion

```
Set variable x = INT((100/_totalframes)*_framesloaded)
```

so wird X auf einen %-Wert der geladenen Frames gesetzt. Soll dieser Wert ständig überprüft werden, muss diese Abfrage natürlich in einer Schleife sein. Wenn Sie auch noch überprüfen möchten, wann er weiterspielen darf, setzt man einfach eine IF-Abfrage:

```
If x = 100
  Goto and Play ("Intro,1 ")
End If
```

So würde er, wenn x 100 ist, also 100 % der Frames geladen sind, zur Szene Intro springen und ab dem ersten Frame spielen.

Damit im Textfeld auch noch das %-Zeichen steht, kann man

```
SetVariable xx = x&"%"
```

einfügen. Die Variable heißt xx, damit die IF-Abfrage davor noch funktioniert: Dort wird überprüft, ob x = »100« ist und nicht »100 %« – das Textfeld muss so natürlich auch die Variable xx darstellen.

Auch diese Version hat wieder den Nachteil, dass die geladenen Frames überprüft werden und nicht die geladenen Bytes.

Eine weitere Flash 4-kompatible Lösung finden Sie auf der CD unter Workshops/DuplicateEffekt.

Preloader mit JavaScript (Flash 3, 4)

In dieser Variante wird tatsächlich die Prozentzahl der geladenen Bytes ermittelt. Allerdings macht das nicht Flash, sondern ein JavaScript. Die ermittelte Variable wird dann an Flash zurückgegeben und dort dargestellt.

Preloader mit Flash 5

Das ist nun sicher der einfachste Weg – es gibt bereits die Variablen.

Allerdings ist noch mehr möglich, und damit wird es wieder kompliziert. Treiben wir es einmal auf die Spitze. Unser Preloader soll Folgendes anzeigen:

- Die verbleibende Zeit
- Die Übertragungsrate
- Die geladenen Bytes
- Die geladenen Prozent
- Einen Ladebalken

Wir brauchen zwei Keyframes. In den ersten Frame kommt das Skript, im zweiten loopen wir zurück zum ersten.

Als Erstes überprüfen wir mit der guten alten Flash 3-Aktion, ob das File nicht bereits gecached ist. Ende ist eine Bildbezeichnung des letzten Keyframes im Film. Start ist der Anfang des Film – wo der geladene Film beginnen soll.

```
ifFrameLoaded ("Ende") {
   gotoAndPlay ("Start");
}
```

Nun wird die Variable `kbytes` definiert. Die Aktion `getBytesLoaded` spricht für sich, sie wird hier nur durch 1024 geteilt, um den Wert in Kilobytes zu erhalten.

```
kbytes = int (this.getBytesLoaded() / 1024) add "KB";
```

Wie auch in der Flash 4-Version mit den Frames werden die geladenen Bytes durch die Bytes insgesamt geteilt und mit 100 multipliziert.

```
x = ((this.getBytesLoaded()/this.getBytesTotal())*100);
```

Die Variable **prozent** soll dargestellt werden. Sie besteht aus der eben ermittelten Zahl (Variable x) und dem %-Zeichen.

```
prozent = int (x) add "%";
```

Im letzten Schritt wird die X-Skalierung des Ladebalkens noch auf die ermittelte Prozentzahl gesetzt.

```
setProperty ("_root.bar", _xscale, x);
```

Die Ermittlung der verbleibenden Zeit setzen wir der Übersichtlichkeit halber in einen eigenen MovieClip. Er besteht aus drei Keyframes.

Im ersten Keyframe ermitteln wir lediglich die bis hierhin verstrichene Zeit und schreiben sie in die Variable **Zeit_1**, um diese Differenz direkt auszugleichen:

```
Zeit_1=int(getTimer()/1000);
```

Im zweiten Keyframe ist das eigentliche Skript:

```
bytes_loaded = _root.getBytesLoaded();
```

Die zu diesem Zeitpunkt geladenen Bytes werden in die Variable **bytes_loaded** geschrieben.

```
bytes_total = _root.getBytesTotal();
```

Die Gesamtgröße des Films in Bytes wird in die Variable **bytes_total** geschrieben.

```
zeit=(getTimer()/1000)-Zeit_1;
```

Hier wird die Zeit seit Filmstart erneut ermittelt. Davon wird **Zeit_1** abgezogen, die Differenz ist für die folgende Rechnung relevant:

```
bps= bytes_loaded/time;
```

Die Rate Bilder pro Sekunde (bps) wird ermittelt.

```
restbytes = bytes_total-bytes_loaded;
```

Die restlichen Bytes.

```
restsek= Math.ceil(restbytes/bps);
```

Die restlichen Sekunden.

```
min=Math.floor(restsek/60);
```

Die restlichen Minuten.

Mit der Aktion `floor` wird der Kommawert abgeschnitten. Dieses Skript formatiert also nur die Ausgabe in eine für uns angenehme Darstellung:

```
if(min>=1){
      restsek-= Math.floor(min*60);
      if (restsek < 10){
         restsek= "0"add restsek;
   }
}
else{
   min="0";
   if (restsek < 10){
      restsek= "0"add restsek;
   }
}
restzeit=min add":"add restsek;
```

Ausgabe der Restzeit im Format »Minuten:Sekunden«.

```
bit_rate = int(bps)add"bps";
```

Ausgabe der Übertragungsrate.

Preloaden von externen Movies

Wenn man externe Movies (mehrere SWFs) verwendet, will man oft sicherstellen, dass diese bereits geladen sind, wenn man sie benötigt. Ohne einen Preloader sind sie aber sofort zu sehen, wenn man sie lädt. Flash 3-kompatibel könnte man im externen Movie in zwei leeren Key-

frames eine "If Frame is Loaded"-Schleife bauen, die, wenn das Movie geladen ist, ein Tell Target an den Hauptfilm zurückgibt: Ich bin geladen – spiele ab.

Ab Flash 4 wird es etwas einfacher:

```
MyMcLoader._visible = false;
loadMovie ("http://www.xyz.de/xyz.swf", "myMcLoader");
```

Im Movie muss man nun – wenn es geladen ist (If _frameloaded=_total frame) – eine Variable setzen, die abgefragt werden kann, wenn das Movie gebraucht wird, um sicherzustellen, dass dieses auch geladen ist. So kann man mit IF-Abfragen z. B. prüfen, »solange der Film noch nicht da ist (»Variable« gleich false), loope diese Abfrage«. Wenn der Film meldet, das geladen ist ("Variable" = true), dann starte ihn. Das ist alles und entspricht im Wesentlichen der Funktion if frame is loaded, hier auf das Laden von externen Movies angewendet.

Ab Flash 4 kann man den Ladezustand des externen Movies auch wie folgt vom Hauptmovie aus abfragen. Wenn es geladen ist, bekommt die Hauptzeitleiste den Befehl play:

```
if ((Number(getProperty("ext_movie", totalframes))>0) and
(Number(getProperty("ext_movie", framesloaded)) ==
Number(getProperty("ext_movie", _totalframes)))) {
   play;
}
```

Nun muss man noch erwähnen, dass es auch eine Möglichkeit gibt, mit JavaScript die geladenen Bytes zu ermitteln und diese Variable an Flash zu übergeben, das ist sicher auch ein sehr genauer Weg, aber seit Flash 5 zu umständlich.

Flash Print

Wo wir schon das Printfeature abfragen können, schauen wir auch, was man damit machen kann. Hier hat Macromedia auf einen User-Wunsch reagiert, da man ihn im Plug-in realisieren konnte.

Carlo Blatz

Wenn das Plug-in mindestens die Version 4.025 hat, kann man in Flash Druckbuttons »bauen« und genau bestimmen, was gedruckt werden soll. Dafür markiert man alle Frames, die gedruckt werden sollen, mit der Bildbezeichnung `#p`. Wenn man einen anderen Bereich drucken möchte als die ganze Bühne, kann man ein Frame mit der Bezeichnung `#b` anlegen und darin eine Fläche zeichnen, die als Druckbereich gewertet wird. Pro Zeitleiste darf es nur ein solches Frame geben. Da es unter Flash 4 noch nicht die Aktion »Print« gab, wird hier der Befehl zum Drucken mit einem GetURL aufgerufen. Es gibt dabei zwei Möglichkeiten, wie Flash drucken soll.

- `GETURL ("print:")` Flash druckt die mit `#p` markierten Frames normal.
- `GETURL ("printAsBitmap:")` Flash druckt die mit `#p` markierten Frames in maximaler Auflösung.

Außerdem gibt es zwei weitere Eigenschaften `#bframe` und `#bmax`. Sie können hinter den Doppelpunkt gehängt werden und überschreiben eventuell gesetzte Frames mit der Bezeichnung `#b`. Der Druckbereich gilt bei dem Aufruf mit `bframe` für das aktuelle Bild, das in maximaler Größe gedruckt wird, bei `bmax` für die maximale Fläche aller druckbaren Bilder als einheitliche Fläche genommen, z. B. um die Proportionen beizubehalten. Die äußersten Kanten der mit `#p` gekennzeichneten Bilder gelten also.

In Flash 5 sind diese Optionen in der Aktion `Print` als Eigenschaften vorhanden. Mit der Eigenschaft Drucken als Vektor wird in maximaler Qualität, aber ohne Transparenzen gedruckt. Bei Drucken als Bitmap werden auch Transparenzen gedruckt. Die Bilddaten werden mit der für den Drucker maximalen Auflösung gedruckt.

Möchte man etwas mitdrucken, das der User auf dem Bildschirm nicht sieht, kann man das mit einer einfachen Filmsequenz machen, die man mit `Set Property _visible` auf `False` setzt. Auch Transformationen wer-

den beim Druck ignoriert. Filmsequenzen müssen sich auf der Bühne bzw. im Arbeitsbereich befinden und einen Instanznamen haben.

Man kann auch das Gegenteil bewirken und das Drucken verhindern. Dabei wird die Option DRUCKEN aus dem Kontextmenü des Flash Players ausgeblendet. Dafür muss das erste Bild der Hauptzeitleiste die Bildbezeichnung `"!#p"` haben.

Mit diesem Know-how kann man nun z. B. einen Banner fertigen, der nur 230 x 33 Pixel hoch ist, auf Knopfdruck aber eine DIN A4-Seite druckt. Dafür erstellt man einfach die Banneranimation. Im Banner befindet sich ein Knopf, der das Drucken auslösen soll. Dieser muss jedoch vorerst überprüfen, ob der User auch ein Plug-in installiert hat, das den Printmodus unterstützt (ab 4.025). Das haben wir bereits unter Flash Detection behandelt, hier aber noch einmal den Code angepasst und für diesen Zweck reduziert:

```
on (release) {
   playerversion = eval("$version");
   versionnumber = substring(playerversion, 9, 2);
   if (Number(versionnumber)>=20) {
      tellTarget ("/print") {
         play ();
      }
   } else {
      tellTarget ("/print") {
         gotoAndPlay ("no_print");
      }
   }
}
```

Der Button springt also mit der Aktion `TellTarget` einen MovieClip an. Dieser MovieClip mit dem Instanznamen `print` liegt außerhalb der Bühne, da er nicht sichtbar sein muss. Dort gibt es einen Keyframe mit der Bezeichnung `no_print`. Hier sollte ein Hinweis erscheinen, dass man sein Plug-in updaten muss.

Der MovieClip ist im ersten leeren Keyframe gestoppt. Wurde das richtige Plug-in gefunden, bekommt der MovieClip die Aktion `Play` und gelangt zum zweiten Keyframe, wo mit der Aktion `LoadMovie` ein externes SWF geladen wird, in dem sich zu druckende Inhalte befinden. Der Aufbau dieses SWFs entspricht den Regeln; die ich oben bereits beschrieben habe (zu druckende Keyframes `#p` und Druckbereich `#b`).

In Frame 3 überprüfen wir, ob der externe Movie **ext_Movie** bereits vollständig übertragen ist:

```
if ((Number(getProperty("ext_movie", totalframes))>0) and
(Number(getProperty("ext_movie", framesloaded)) ==
Number(getProperty("ext_movie", _totalframes)))) {
   gotoAndPlay("Print");
}
```

In Frame 4 loopen wir zurück zu Frame 3. Der Loop wird verlassen und zu Frame **"Print"** gesprungen; wenn der externe Movie geladen ist. Dort wird der Druck der externen Datei initiiert:

```
getURL ("print:", "tech_sheet");
```

Das war bis hier Flash 4_kompatibel. In Flash 5 kann man den Preloader und die Flash Detection etwas optimieren sowie den Aufruf des Printbefehls durch die Aktion **Print** lösen:

```
print ("ext_movie", "bmovie");
```

Da ein Banner aber für möglichst viele User funktionieren soll, genügt der Flash 4-Code.

Cookies anlegen und auslesen

Cookies sind Werte in ASCII-Dateien, die durch die aufgerufene HTML-Seite mittels JavaScript generiert und dem Browser zum Speichern auf der Festplatte des Anwenders übergeben werden.

Cookie anlegen und Wert setzen

▶ Martin Fleck
www.action-script.com

Verwenden Sie Einfügen·Neues Symbol (Strg+F8), um eine neue Filmsequenz einzufügen. Im darauf folgenden Dialog vergeben Sie als Namen »Willkommen«. Klicken Sie das Werkzeug Text in der Palette Tools an, um ein Textfeld auf der Bühne zu erstellen. Öffnen Sie über Fenster·Bedienfelder das Bedienfeld Texteigenschaften. Markieren Sie das Textfeld, und wählen Sie aus der Drop-down-Liste der Palette den Eintrag Input Text, um dem Anwender das Eingeben seines Namens zu ermöglichen. Vergeben Sie als Variable `"Name"`, um auf den Inhalt des Textfeldes zugreifen zu können.

Mit (Strg+F8) sollten Sie eine neue Schaltfläche einfügen. Geben Sie anschließend als Namen »OK« ein, und aktivieren Sie den Punkt Schaltfläche. Nun über das Textwerkzeug den Schriftzug OK erstellen. Wechseln Sie zurück in den Bearbeitungsmodus der Filmsequenz »Willkommen«.

Öffnen Sie die Bibliothek (Strg+L), die alle Elemente des Films beinhaltet, und ziehen Sie die Schaltfläche »OK« neben das Textfeld. Klicken Sie mit der rechten Maustaste auf die Schaltfläche, und wählen Sie anschließend den Menüeintrag Aktionen. Das folgende ActionScript wird ausgelöst, sobald der Anwender auf die Schaltfläche klickt.

```
on (release) {

getURL ("javascript:zeit = new Date(); zeit.setTime(zeit.getTime()
 + (10*24*60*60*1000)); document.cookie = 'Name=" + Name + ";
expires=' + zeit.toGMTString()void(0);")
}
```

Über die `GetURL`-Funktion werden JavaScript-Anweisungen aufgerufen, die den aktuellen Inhalt des Textfeldes in einem Cookie speichern. Der Cookie bleibt zehn Tage auf dem Rechner des Anwenders gespeichert. Mit der Anweisung, die mit `document.cookie =` beginnt, wird der Cookie geschrieben.

Wechseln Sie nun mit (Strg+E) zum Hauptfilm zurück. Ziehen Sie die Filmsequenz »Willkommen« aus der Bibliothek (Strg+L) auf die Bühne, und vergeben Sie als Instanznamen »Willkommen«.

Cookie auslesen und gespeicherten Wert ermitteln

Um einen Wert des Cookies zu ermitteln, müssen Sie Ihn vorher auch gesetzt haben. Beim Einbinden des Filmes in die HTML-Seite müssen Sie darauf achten, dass Sie dem Film auch einen Bezeichner wie `movie` geben. Im Explorer geschieht das über das Einbinden des ID-Tags und in Netscape über das Name-Tag. Erstellen Sie eine neue Ebene; und wählen Sie als Bezeichnung »Aktionen«. Fügen Sie in der aktuellen Ebene folgendes ActionScript in das erste Schlüsselbild ein:

```
getURL ("javascript:document.movie.SetVariable
('Willkommen.Cookie', document.cookie);");
```

Der Inhalt des Cookies wird in die Variable `Cookie` der Filmsequenz »Willkommen« abgespeichert.

Fügen Sie an Position 3 des Films ein Schlüsselbild (F6) ein. Setzen Sie die Bildrate des Films auf 1, um das Übertragen des Cookies aus JavaScript in die Variable `Cookie` erfolgreich durchzuführen. Klicken Sie nun mit Doppelklick auf das Schlüsselbild, und geben Sie folgendes ActionScript ein:

```
with (Willkommen) {
```

Der Zielpfad für Aktionen wird geändert, alle Vorgänge beziehen sich auf die Filmsequenz »Willkommen«.

```
   if (length(Cookie) > 0) {
```

Sollte die Länge der Variablen `Cookie` größer als 0 sein, wird der kommende Anweisungsblock ausgeführt.

```
    Position = Cookie.indexOf("=");
```

Durchsucht den Wert **Cookie** nach dem Zeichen »=« und speichert dessen Position in der Variablen **Position**.

```
    Willkommen.Name = Cookie.Substring(Position + 1);
```

Zeigt den Inhalt der Variablen **Cookie**, der nach dem Gleichheitszeichen kommt, im Textfeld an.

```
  }
```

Ende des Anweisungsblocks von If.

```
  else {
```

Sollte die Länge der Variablen **Cookie** nicht größer als 0 sein, wird der kommende Anweisungsblock ausgeführt.

```
    Willkommen.Name = "nix eingegeben";
```

Zeigt den Text »nix eingegeben« im Textfeld an.

```
  }
```

Ende des Anweisungsblock von Else.

```
}
```

Ende der With-Anweisung.

```
stop();
```

Stoppt den Film.

Fügen Sie an Position 3 der Ebene, in der sich die Filmsequenz »Willkommen« befindet, ein leeres Schlüsselbild F5 ein.

Mausposition und -verfolgung in Flash 4 und 5

Oft sieht man auf diversen Seiten, wie Objekte scheinbar magnetisiert der Maus folgen, obwohl die direkte Abfrage der Mausposition in Flash 4 nicht möglich ist. Für diesen und verschiedene andere Effekte ist es aber unerlässlich, die Position der Maus zu kennen.

▸ Stephan Fischer
www.phaetons.de

Mein Ziel ist es, die Grundlagen zur Positionsermittlung und Mausverfolgung möglichst einfach zu erklären. Darauf aufbauend sollten sich einige andere Effekte verwirklichen lassen. Den Umgang mit Filmsequenzen sollte man bereits beherrschen.

Ich habe ein Kapitel zu Flash 5 angehängt, da ich aber denke, dass viele Flash 4 noch eine Weile benutzen werden, habe ich den alten Teil so belassen.

Ermitteln der Mausposition – Lösungsansätze in Flash 4

Um die Mausposition in Flash 4 feststellen zu können, gibt es meines Wissens zwei Ansätze. Einen einfacheren, der aber auch ungenauer ist bzw. Fleißarbeit bedeutet, wenn er genau sein soll, und auf der anderen Seite die genaue Ermittlung der Position durch einen kleinen Trick.

1. Lösung: unsichtbare Sensoren

Diese Lösung beruht darauf, auf der betreffenden Fläche, in der die Mausposition abgefragt werden soll, viele kleine unsichtbare Buttons anzulegen. Versieht man diese Buttons mit einer Rollover-Aktion, weiß man auch, an welcher Stelle sich die Maus befindet.

Der Vorteil liegt darin, dass die Buttons einfach zu erstellen sind, man die Position nicht ständig abfragen muss (durch die Rollover-Aktion meldet sich der Button von allein) und dass man die Form der »Sensorflächen« vollkommen frei bestimmen kann.

Der Nachteil liegt darin, dass man je nach gewünschter Genauigkeit der Abfrage auch die Anzahl der Buttons erhöhen muss. Dies kann einen so hohen Arbeitsaufwand erfordern, dass man über Lösung 2 vielleicht einfacher an sein Ziel kommt. Des Weiteren verändert sich der Mauszeiger zur Hand, sobald er über einen Button bewegt wird, dieses ist in einigen Fällen auch störend.

2. Lösung: angehängtes unsichtbares Symbol

In diesem Fall wird beim Starten des Flash-Films (oder bei Bedarf auch an anderer Stelle) ein unsichtbares Symbol an den Mauszeiger gehängt, dessen Position man jederzeit abfragen kann, womit man auch gleichzeitig die Position der Maus erhält (in Flash 5 geht das einfach mit `_xmouse` und `_ymouse`).

Hier die einzelnen Schritte.

1. Schritt:
   Man erzeuge ein neues Symbol ohne Inhalt mit der Eigenschaft Filmsequenz und gebe dem Ganzen einen Instanznamen (im Beispiel `unsichtbarer_clip`).
2. Schritt:
   Dieses Symbol wird jetzt an den Mauszeiger gehängt, indem man in das erste Frame der Zeitleiste unter Eigenschaften/Aktionen folgende Aktion einträgt : `Start Drag ("unsichtbarer_clip")`
   Die Option Lock mouse to center sollte aktiviert werden, damit man die genauen Koordinaten erhält.
3. Schritt:
   Nun legt man im ersten Frame des Hauptfilmes die Variablen `position_x` und `position_y` an. Dies geht mit der Aktion:
   `Set Variable: "position_x" = ""`
   und
   `Set Variable: "position_y" = ""`
   Die Werte kann man vorerst leer lassen.
4. Schritt:
   Erstellen Sie ein MovieClip-Symbol für die Positionsabfrage. (Ich erzeuge für die Abfrage absichtlich ein eigenes Symbol und setze dieses in eine eigene Ebene. Somit ist die Abfrage unabhängig von dem Rest

des Films aktiv. Sie kann aber nach Bedarf auch an anderer Stelle eingesetzt werden). Erzeugen Sie dann ein neues Symbol mit zwei Frames. In das erste Frame kommt die Abfrage, in das zweite Frame ein Goto-Frame (Genaueres zur Abfrage finden Sie in Beispiel 2).

Verfolgen der Maus mit Objekten in Flash 4

Das Prinzip: Wenn man erst einmal die Position ermittelt hat, kann man einem anderen Objekt über `SetProperty` sagen, auf welche Position es sich bewegen soll oder wie dieses Objekt reagieren soll (z. B. skalieren oder rotieren).

Im Beispiel 1 wird über den weiter oben beschriebenen ersten Lösungsansatz ein Uhrzeiger nach dem Mauszeiger ausgerichtet.

Im zweiten Beispiel wird die Position über den zweiten Lösungsweg ermittelt (angehängtes Objekt), und ein anderes Objekt verfolgt über `SetProperty` den Mauszeiger.

Beispiel 1 – Uhrzeiger

Wenn man den Mauszeiger über das Ziffernblatt der Uhr bewegt, richtet sich der Minutenzeiger nach der Maus aus. Der Mauszeiger ist hier ein einzelnes Symbol, als Filmsequenz definiert mit dem Instanznamen »stunde« (nur Filmsequenzen lassen sich mit `SetProperty` ansprechen, mit Buttons bzw. Grafiken geht dieses nicht).

Als »Mausdetektor« habe ich einen Button erzeugt, der den Bereich einer Stunde auf dem Ziffernblatt abdeckt. Diesen habe ich dann für jede weitere Stunde kopiert und entsprechend der benötigten Position um jeweils 30° rotiert und richtig ausgerichtet.

Damit der Stundenzeiger der Maus folgt, habe ich die Buttons mit folgender Aktion belegt:

```
On (Roll Over)
  SetProperty("/stunde", Rotation) = "0"
End On
```

◀ Abbildung 1
Uhr

`"/stunde"` steht hierbei für den Instanznamen des Symbols, Rotation natürlich für das Rotieren. Die 0 steht für 0°-Rotation (angefangen bei Stunde 12). Entsprechend habe ich für jeden weiteren Button bzw. jede weitere Stunde 30° addiert (360° : 12 = 30°). Das war schon alles!

Ebenso könnte man das Gleiche für den Minutenzeiger machen, wobei hier dann schon 60 Buttons erstellt werden müssten und die Rotation pro Minute

360° : 60 = 6° entsprechen würde. Das Ganze wäre schon etwas mehr Aufwand, was auch sehr gut zeigt, welchen Nachteil diese Methode eventuell hat.

(Eine andere Möglichkeit wäre auch noch, eine Filmsequenz mit entsprechend vielen Frames für jede Zeigerstellung zu erstellen und mit den Buttons die Filmsequenz in das passende Frame springen zu lassen. Dazu benötigt man den TellTarget-Befehl, dessen Erläuterung hier aber den Rahmen sprengen würde.)

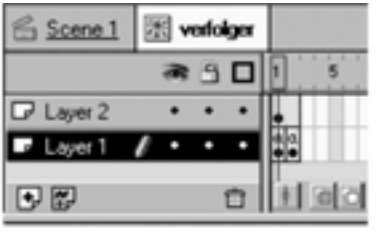
Beispiel 2 – Objekt folgt dem Mauszeiger

In diesem Beispiel wird, wie in Lösungsansatz 2 beschrieben, die X- und Y-Position ermittelt und in den Variablen `position_x` und `position_y` gespeichert.

Das Symbol, das dem Mauszeiger folgen soll, wurde als Filmsequenz definiert (nur hier kann man die Position mit `SetProperty` verändern). Der Instanzname heißt `verfolger`.

Die Aktionen für die Bewegung des Verfolgers habe ich mit in das gleiche Symbol geschrieben (theoretisch könnten sie aber auch in der Hauptzeitleiste oder einem anderen Symbol liegen). Damit die Positionen ständig verglichen werden, habe ich zwei Frames angelegt, wobei im zweiten Frame die Anweisung `"Go to and Play (1)"` steht, damit ein Loop entsteht.

Abbildung 2 ▶
Zeitleiste des Symbols »verfolger«



In Frame 1 steht das wichtige ActionScript als Erstes:

```
If (_level0/:position_x<verfolger_x)
  Set Property ("/verfolger", X Position) = verfolger_x-1
End If
```

Hiermit bekommen die Variablen `verfolger_x` und `verfolger_y` immer die aktuellen X- und Y-Koordinaten des Verfolgersymbols zugewiesen. Normalerweise kann man auch im weiteren Skript direkt mit `GetProperty ("/verfolger",_x)` arbeiten, wenn man aber bedenkt, dass eventuell noch sehr viele Vergleiche gemacht werden müssen, ist das Arbeiten mit Variablen (die dann kurz vielleicht nur noch `vx` und `vy` heißen) wesentlich einfacher.

Damit sich der Verfolger auch noch bewegt, kann man folgende Aktion benutzen:

```
If (_level0/:position_x<verfolger_x)
  Set Property ("/verfolger", X Position) = verfolger_x-1
End If
```

Das Gleiche macht man dann noch einmal mit `position_y` und `verfolger_y`, und man ist fertig.

Maus- bzw. Objektverfolgung in Flash 5

Die neue Version von Flash bringt wesentliche Erleichterungen, was das Skripten angeht. Ursprünglich wollte ich nur das vorhandene Flash 4-Skript an Flash 5 anpassen. Beim Probieren merkte ich aber, dass man mit objektorientierter Programmierung ziemlich einfach ein Objekt fertigen

kann, das individuell der Maus oder auch einem anderen Objekt folgen kann. Das will ich Ihnen natürlich nicht vorenthalten, Sie können es in Beispiel 3 sehen.

Objektorientiertes Skripten

Vereinfacht bedeutet es, dass man z. B. Funktionen, die öfter (von verschiedenen Programmteilen) aufgerufen werden, auch als einzelne Funktion programmiert und nicht wiederholt an verschiedenen Stellen eingibt. Diese Funktion kann man dann von allen anderen Stellen aufrufen, und man spart somit Speicherplatz im Quellcode.

In diesem Beispiel bedeutet es, dass ich eine Filmsequenz habe, die ein eigenes Skript ausführt und somit ihre Position selbst bestimmt. Dadurch, dass ich bestimmte Befehle/Variablen auf das Objekt (die Filmsequenz) selbst beziehe, kann jede Instanz dieses Objektes eigenständig arbeiten.

Beispiel 3 – Mausverfolgung mit Flash 5

Wie in Abbildung 3 zu sehen, besteht das Beispiel nur noch aus einem Symbol, das ich `verfolger` genannt habe. Die Filmsequenz `radar` ist nur Bestandteil von `verfolger` und enthält die Animation des Radars, ist aber für die Funktion selbst unwichtig.

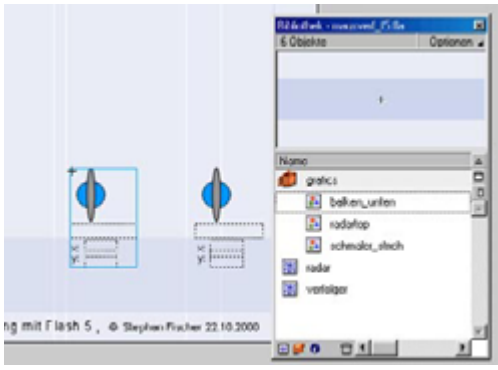
Von dem Symbol `verfolger` habe ich zwei Instanzen auf der Bühne platziert und die Instanznamen `verfolger1` und `verfolger2` vergeben. Die Instanz `verfolger1` soll der Maus folgen; und Instanz `verfolger2` soll `verfolger1` folgen.

Damit das Objekt weiß, was es machen soll, setze ich ein paar Variablen in die Objektaktionen der entsprechenden Instanz. Dorthin kommt man, indem man die Instanz mit der rechten Maustaste anklickt (auf dem Mac: Ctrl-Taste und dann Mausklick) und dann auf Aktionen klickt.

Das Skript sieht folgendermaßen aus:

```
onClipEvent (load) {
   xpos1 = this._x;
   ypos1 = this._y;
   folge = maus;
   abstandx = 0;
   abstandy = 0;
}
```

Abbildung 3 ▶
Bühne und Bibliothek
mit den Symbolen



Beim Laden der Filmsequenz (`onClipEvent (load)`) werden Variablen definiert, die sich auf das Objekt beziehen. `xpos1` und `ypos1` entsprechen der aktuellen X- und Y-Position der Filminstanz beim Start. Das Schlüsselwort `this`. bezieht sich immer auf das Objekt, in dem der Code steht.

Die Variable `folge` muss auf `maus` gesetzt werden, wenn die Filminstanz der Maus folgen soll. Soll die Filminstanz einem Objekt folgen, muss der absolute Pfad der zu verfolgenden Filminstanz angegeben werden. In den Objektaktionen der Instanz `verfolger2` habe ich entsprechend `folge = "/verfolger1"` gesetzt.

Als Letztes setze ich die Variablen `abstandx` und `abstandy`, die den Abstand zu dem verfolgten Objekt oder der Maus in Pixeln angeben. Statt einer Zahl kann man hier natürlich auch einen Ausdruck benutzen, zum Beispiel die Höhe des Objekts (`abstandx=this._width;`), wie ich es in der Instanz `verfolger2` getan habe.

Das sind die einzigen Angaben, die für jede Filminstanz getrennt gemacht werden müssen. Man bedenke, dass man hier nicht das Ursprungssymbol verändert, sondern nur die jeweiligen Instanzen!

Als Nächstes kommt das Skript, das für die Bewegung zuständig ist. Es steht in der Zeitleiste der Filminstanz und ist verständlicherweise in jeder Instanz gleich. Es muss bzw. darf nicht abgeändert werden (außer für Testzwecke natürlich)! Die Zeitleiste besteht aus zwei Frames, damit ich eine ständig laufende Schleife erzeugen kann.

Skript in Frame 1:

```
if (this.folge == maus) {
   Xpos = _level0._xmouse;
```

◀ **Abbildung 4**
Zeitleiste der Filmsequenz »verfolger«

```
   Ypos = _level0._ymouse;
   } else {
   Xpos = this.folge add ._x;
   Ypos = this.folge add ._y;
}
```

Hier prüfe ich, ob die Filminstanz der Maus folgen soll oder einer anderen Filminstanz, und bestimme die aktuelle Sollposition (wohin die Instanz folgen soll). Ist die Variable `folge` dieser Instanz auf `maus` gesetzt, belege ich die Variablen `Xpos` und `Ypos` mit der aktuellen Position der Maus (`_xmouse` und `_ymouse`).

Ich setze _level0. davor, damit ich den absoluten Wert von der oberen linken Bühnenecke aus bekomme. Ohne dies bekommt man den Wert relativ zur Filminstanz, der in diesem Fall nicht weiterhilft.

Steht etwas anderes in der Variablen `folge`, gehe ich davon aus, dass eine Filminstanz angegeben ist, und hole mir die X- und Y-Position von dieser Filminstanz.

```
xpos1 = (Xpos+3*xpos1)/4;
ypos1 = (Ypos+3*ypos1)/4;
this._x = Math.floor(xpos1+abstandx);
this._y = Math.floor(ypos1+abstandy);
```

Als Nächstes berechne ich die neue Position der Filminstanz und weise sie den Variablen `xpos1` und `ypos1` zu. Die Formel `(Xpos+3*xpos1)/4` ergibt eine Teilstrecke zwischen den beiden Objekten (Instanz/Maus oder Instanz/Instanz), um die die verfolgende Filminstanz verschoben wird. Diese Teilstrecke wird prozentual immer kleiner, wodurch der Verzögerungseffekt entsteht.

Mit `this._x` und `this._y` setze ich dann die Position der aktuellen Filminstanz neu, welche aus der nächst kleineren ganzzahligen (Math.floor) Summe aus `xpos + abstandx` bzw. `ypos + abstandy` besteht.

Skript in Frame 2:

```
gotoAndPlay (1);
```

Hiermit erzeuge ich eine ständig laufende Schleife, damit die Position auch immer wieder neu berechnet wird.

Das war schon alles!

Wie man sieht, kann man mit einem kleinen Skript große Wirkung erzielen, der komplette Film hat gerade 6 KB (mit eingebundener Schriftart 23 KB).

Grafiken der Maus folgen lassen

▸ boblgum
www.mysterion.de

Hier möchte ich Ihnen eine weitere Möglichkeit zeigen, wie man Grafiken der Maus folgen lassen und somit beeindruckende Effekte erzielen kann. Hierbei möchte ich auch auf einige neue Funktionen in Flash 5 eingehen.

Was wir brauchen:

- Eine Grafik, die der Maus folgen soll
- Einen Soundloop

Ziel

Die Grafik soll der Maus folgen. Es soll sich ein »Schwanz« hinter der Grafik bilden. Der Soundpan (in welcher Box (links oder rechts) der Sound zu hören ist), die Soundlautstärke und die Farbe der Grafik sollen sich in Abhängigkeit der Mausposition ändern. Und zwar:

- Oben: der Sound wird leiser
- Links: der Sound »wandert« in die linke Box
- Rechts: der Sound »wandert« in die rechte Box

Grafik

Als Erstes zeichnen wir irgendeine Grafik, von der wir überzeugt sind, dass die Welt noch nie so einen prima Maustrail gesehen hat. Wir wandeln sie sofort in einen MovieClip mit dem Instanznamen `grafik` um.

Wie immer habe ich mindestens zwei Ebenen angelegt. Dabei ist die unterste Ebene immer für ActionScript zuständig.

Nun wenden wir uns dem ActionScript im ersten (und einzigen) Keyframe der Hauptzeitleiste zu.

```
Mouse.hide();
i = 1;
do {
   duplicateMovieClip (_root.grafik, i, i);
   i++;
} while (i<50);
```

Wir blenden den Mauszeiger aus und duplizieren unsere Grafik 50 Mal.

Bei dieser Methode wird das ActionScript in den MovieClip geschrieben, so dass jeder MovieClip sich selbst positioniert. Das setzt natürlich voraus, dass jeder der MovieClips ein Merkmal hat, das ihn von allen anderen unterscheidet und mit dem wir rechnen können. Hier bietet sich der Name der Instanz an. In unserem Fall wären es Werte von 1 bis 50, da wir das »Originalsymbol« nicht direkt verwenden, sondern nur Duplikate davon.

Nun zur Zeitleiste des MovieClips und dem darin enthaltenen ActionScript. In der ActionScript-Ebene haben wir zwei Keyframes. Den zweiten Keyframe brauchen wir, um eine Schleife zu erzeugen. Hierzu benutzen wir einfach:

```
   gotoAndPlay(_currentframe-1);
```

Im ersten Keyframe wird es schon etwas interessanter.

Hier ist meine Methode, um ein Objekt auf einen Punkt (die Maus) »zufliegen« zu lassen, so, dass sich die Geschwindigkeit mit der zunehmenden Annäherung an die Zielkoordinate verringert. Kommen wir also zum ActionScript im Keyframe 1.

Nur die Zeilen 2 und 3 sind für die Bewegung zu dem Mauszeiger zuständig. Alles andere ist nur Spielerei. Gehen wir Zeile für Zeile durch.

```
if (_x<>_root. _xmouse or _y<>_root._ymouse) {
```

Hier wird abgefragt, wo sich die Grafik gerade befindet. Wenn die Koordinaten der Grafik nicht mit den Koordinaten des Mauszeigers übereinstimmen, werden die Anweisungen ausgeführt.

```
_x=_x-(_x-_root._xmouse)/50*_name;
```

Es wird die X-Koordinate der Grafik festgelegt, indem die Differenz zwischen Grafik und Maus durch eine feste Zahl (`50*_name`) geteilt und diese von der aktuellen Position abgezogen wird. Die Abstände zwischen den Werten werden immer kleiner, ergo die Bewegung langsamer. Wenn man nicht durch 50, sondern durch eine kleinere Zahl dividiert, wird die Bewegung dementsprechend schneller. Damit sich jeder der durch Duplikation erzeugten MovieClips anders bewegt, multiplizieren wir den Wert 50 mit `_name`, weil er ja bei uns eine Zahl ist und von Flash auch so gehandhabt wird.

```
_y=_y-(_y-_root._ymouse)/50*_name+(_root._ymouse<200)
*5-(_root._ymouse>=200)*5;
```

Hier hat sich nur geändert, dass wir den »Schwanz« in Abhängigkeit von der vertikalen Mitte der Bühne positionieren. Wenn der Mauszeiger über der Mitte ist, werden die MovieClips nach unten hin ausgerichtet und umgekehrt.

```
_xscale=5*_name;
_yscale=5*_name;
_rotation=(_x-_root._xmouse);
```

Die Zeilen 4, 5 und 6 sind für das Erscheinungsbild der einzelnen MovieClips zuständig (Größe und die Rotation), was man wohl unschwer erkennen kann.

```
}
```

Hier kann man natürlich einiges verändern, um die Bewegung eigenen Wünschen und Vorstellungen anzupassen. Das sei aber der Experimentierlust des Einzelnen überlassen.

◄ Abbildung 5
Soundeinstellungen

Sound

Gehen wir aber noch einen Schritt weiter und »hängen« einen Soundloop an die Maus. Was wir brauchen ist natürlich eine Sounddatei, die wir in Flash importieren. Dann legen wir einen neuen MovieClip »sound« an. Dieser braucht nur eine Ebene und einen Keyframe, wo der Sound liegen soll.

Um den Sound in Flash »ansprechen« zu können, müssen wir ein SoundObject anlegen (siehe Seite 256). Das machen wir (wie auch die restlichen Anweisungen) in den Objektaktionen. Sehen wir uns die Anweisungen genauer an.

```
OnclipEvent (load) {
   Sound=new Sound ();
}
```

Hier wird ein neues SoundObject angelegt. Da unsere Anweisungen im gleichen MovieClip sind, wo auch der Sound ist, steht in der Klammer hinter `"new Sound"` nichts. Wäre es nicht der Fall, so müsste man den MovieClip angeben, in dem der zu manipulierende Sound liegt.

```
OnclipEvent (enterFrame) {
   Sound.setPan(((_root._xmouse/2.75)-100));
   Sound.setVolume((_root._ymouse/4+10));
}
```

Hier manipulieren wir die Balance (links/rechts) und die Lautstärke des Sounds. Die Werte in der Klammer (Argumente) können/dürfen dabei zwischen –100 und 100 liegen. Deswegen nehmen wir nicht direkt die

Mauskoordinaten, sondern rechnen sie so um, dass sie der Größe der Bühne entsprechen.

Die Bühne meines Films ist 550 px breit. Die Mitte wäre dann bei 275. Zwischen x = 0 und x = 275 müssen 100 Einheiten passen. Eine Einheit wäre dann 2,75 (275/100) groß. Um auf –100 bei x = 0 zu kommen, dividiere ich die X-Koordinate und subtrahiere 100. Die Rechnung geht auch bei allen anderen Werten auf. Beinahe dieselbe Rechnung gilt auch für die Lautstärke des Sounds. Da meine Bühne 400 px hoch ist, teile ich die Y-Koordinate der Maus durch 4. Damit ich aber bei y = 0 wenigstens etwas hören kann, addiere ich 10 dazu. Manch einer wird sich wohl fragen, warum wir hier nicht wieder mit einer Zwei-Frames-Schleife arbeiten. Nun, in diesem Fall ist es nicht nötig. Es scheint so, als ob der Film nie tatsächlich stehen bleibt, wenn man nicht gerade eine Stoppaktion definiert hat.

Mit dem Sound wären wir dann auch fertig.

Farbe des Maustrails

Wir verändern auch die Farbe des Maustrails abhängig von den Mauskoordinaten. Dazu müssen wir nur Folgendes durchführen.

Als Erstes erweitern wir den Aufbau unserer Grafik. Wir hatten bisher zwei Frames in jeder Ebene. Wir brauchen aber noch einen zusätzlichen Frame. Dazu verschieben wir die Frames der ActionScript-Ebene einfach um einen Frame nach rechts, und in der Grafikebene fügen wir ein Bild hinzu (F5).

Genau wie bei dem Sound, so müssen wir auch bei der Farbe zuerst ein Farbobjekt anlegen. Das machen wir im ersten Keyframe der ActionScript-Ebene mit:

```
color = new Color(this);
```

Seltsamerweise müssen wir hier das Ziel in der Klammer angeben.

Den zweiten Frame erweitern wir auch um drei Zeilen Code.

```
colorTrans = new Object ();
colorTrans ={ ra: (_x/5), ga: (_y/3), ba: _name*2}
color.setTransform(colorTrans);
```

Hier arbeiten wir wieder mit einem Objekt. Dieses Objekt beinhaltet die Farbtransformationswerte, die wir dann in der letzten Zeile dem Farbeobjekt zuweisen. Wie in der Online-Hilfe erklärt ist, haben wir die Möglichkeit, die Alpha- und die Offsetwerte der Grundfarben (Rot, Grün, Blau) zu manipulieren. In diesem Beispiel möchte ich nur die Alphawerte manipulieren, damit das Ganze nicht noch mehr an Übersichtlichkeit verliert. Die Alphaeigenschaft (Transparenz) der Grafik will ich auch nicht verändern, obwohl es auch nicht weiter kompliziert wäre. Was machen wir nun in der zweiten Zeile von unten genau? Wir koppeln den Alphawert der roten Anteile der Farbe der Grafik (ra) an die X-Koordinate der Grafik, die grünen Anteile an die Y-Koordinate und die blauen Anteile an die Größe des jeweiligen MovieClips, weil wir in Flash eigentlich nur zwei Dimensionen haben (X und Y). Die Umrechnung der Koordinaten erfolgt ganz genau so wie bei dem Sound-Object.

Natürlich könnte man das alles auch anders machen, um andere Farben zu erhalten. Hauptsache ist nur, dass man nicht mit einer schwarzen Grafik arbeitet, weil man da ziemlich schlecht die Farben beeinflussen kann, wenn überhaupt. Am einfachsten ist es, wenn man die Grafik weiß einfärbt, weil ja in der Farbe Weiß alle anderen Farben schon enthalten sind und man sie nur zu manipulieren braucht.

Dynamisches Menüsystem erstellen

Im Folgenden beschreiben wir, wie man ein sehr einfaches, aber schickes Menüsystem hinbekommt, ohne großartige Mathekenntnisse zu besitzen.

Ablauf

Fangen wir mit dem Aufbau des Flashfiles an: Dazu brauchen wir eine Szene und drei Bilder in der Hauptzeitleiste. Die Framerate stellen wir auf 25 und die Bühnengröße auf ca. 600 x 200.

Jetzt erstellen wir erst einmal einen Button. Der kann einfach eckig sein und eine beliebige Farbe haben. Schöner ist es natürlich, wenn Sie sich ein wenig Mühe geben, und noch die Bilder »darüber« und »gedrückt« verändert. Dieser Button ist eigentlich schon unser einziges Objekt. Er reicht, um das komplette Menüsystem aufzubauen.

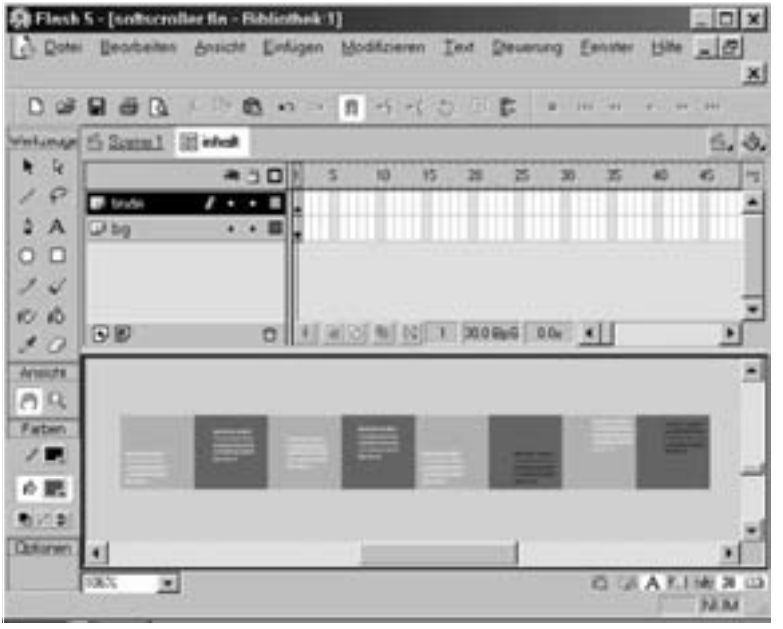
Nun wird es ein wenig komplizierter. Wir erstellen nämlich jetzt das zweite Objekt. Es ist eine einfache Grafik. Im Hauptfilm drücken Sie also einfach nur Strg+F8. Diese Grafik wir unser Menüinhalt sein.

Diese Grafik besteht aus beliebig vielen Ebenen und muss exakt immer einem Vielfaches der Filmbreite (hier 600 Pixel) entsprechen. Das ist natürlich schwierig, da Flash ja gar nicht so viel Platz anbietet. Also genehmigen wir uns einen kleinen Trick. Die Überlegung ist, dass Flash ja skalieren kann wie ein Weltmeister. Es ist ja überhaupt nicht nötig, hier den ganzen Platz zu verbrauchen. Also nehmen wir zur Übersicht einen Skalar, der unter 1 liegt (das heißt, einfach verkleinern; muss noch nicht einmal proportional sein).

Wir bauen uns jetzt den kompletten Inhalt zusammen und stellen ihn nebeneinander auf die Bühne (siehe Abbildung 1).

Der Mittelpunkt steht, der Einfachheit halber, oben links. Ich habe hier z. B. acht Seiten Content und somit 480 Pixel Breite (klar, einfach durch 10 gerechnet). Damit kann man ganz gut umgehen.

Geben wir jetzt ein wenig ActionScript ein. Das ist aber wirklich diesmal ganz einfach.



◀ **Abbildung 1**
Die Bühne mit dem kompletten Inhalt

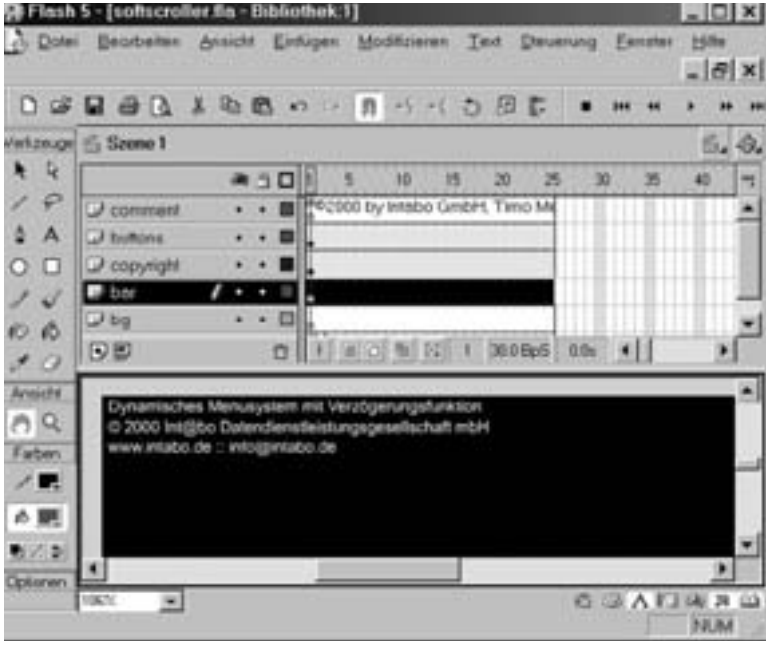
Wie gesagt, ich habe nur drei Bilder im Hauptfilm (siehe Abb. 2). In jedem Bild ist ein wenig ActionScript versteckt.

Bild 1

```
fscommand ("fullscreen", "false");
   freeze =5;
breite =400;
hoehe =200;
mx =0;
my =0;
_root.bar._xscale = 1000;
_root.bar._yscale = 290;
```

`bar` ist der Name von der Menügrafikinstanz. `Mx` und `my` deuten darauf hin, dass man nicht nur ein horizontales Scrolling damit bewerkstelligen kann.

Abbildung 2 ▶
Die drei Bilder des Hauptfilms



Die Skalierung von 1000 bestimmt die genaue Größe der Instanz. Die 290 ist ein grober Schätzwert.

Bild 2

```
barx = _root.bar._x;
bary = _root.bar._y;

If (mx > barx) {
   _root.bar._x = barx +
((mx-barx)/freeze);
}
If (mx < barx) {
   _root.bar._x = barx - ((barx-mx)/freeze);
}
If (my > bary) {
   _root.bar._y = bary + ((my-bary)/freeze);
```

```
}
If (my < bary) {
   _root.bar._y = bary - ((bary-my)/freeze);
}
```

Hier wir anhand des Wertes `mx` die Position der Instanz bar immer wieder angepasst. Dabei gibt der Wert freeze die Verzögerung an. Hier sieht man auch gut, dass man genauso den Wert `my` anpassen könnte.

Bild 3

```
gotoAndPlay (_currentframe-1);
```

Wohin Sie den MovieClip `Menü_` speichern, ist egal, da es – siehe oben – angepasst wird.

Dann fehlt eigentlich nur noch der Button ActionScript. Da wir im Hauptfilm eine Endlosschleife haben, fällt es uns nicht schwer, den zu generieren. Er beschränkt sich auf:

```
on (release) {
      mx = -1800;
}
```

Hier gebe ich ein Vielfaches der Filmbreite an. 1800 / 600 = 3. Richtig! Das ist der Code von Knopf Nummer vier.

Statt `on (release)` kann man natürlich auch ein anderes Event annehmen.

Anhang

646 FAQ

646 Häufige Fragen rund um Flash

650 Was Flasher noch brauchen

658 Wishlist Flash 6

660 Compiler-Fehlermeldungen

664 Webstatistiken

670 HTML 3.2-ASCII-Zeichen

678 Die Autoren dieses Buches

682 Die CD-ROM

FAQ

Frequently asked Questions

Für Flashworker habe ich auf einer Seite alle häufig gestellten Fragen und Antworten gesammelt. In der Newsgroup macromedia.general.germany sammeln sich seit vielen Jahren Flasher. Da hier auch oft neue Teilnehmer hinzukommen, werden gleiche Fragen oft sogar mehrmals in einer Woche gestellt. Die wichtigsten, die nicht bereits im Buch behandelt wurden, möchte ich hier einmal zusammenstellen.

Häufige Fragen rund um Flash

Wie kann man aus Flash eine E-Mail mit vordefiniertem Subjekt senden?

Mit dem Befehlt `GETURL` kann man nicht nur Webseiten ansteuern, sondern auch E-Mail-Adressen. Statt einer URL wird »mailto:empfaenger@adresse.de« eingetragen! Wenn die E-Mail bereits ein Subjekt (Betreff) vorgegeben haben soll, müsste der Befehl wie folgt lauten:

```
mailto:empfaenger@adresse.de?subject=betreffzeile
```

Wie kann man zu den internen Bibliotheken etwas hinzufügen?

Dafür kann man einfach einen neuen Film erstellen und alle Symbole importieren oder erstellen, so dass sie sich in der Bibliothek befinden. Dieses File speichert man im Flash-Verzeichnis als *.fla in den Ordner Bibliotheken/Libraries, und nach dem Schließen des *.fla zeigt es sich im Menü Fenster · Bibliotheken.

Wie kann man ein Movie rückwärts laufen lassen?

Entweder sieht man eine Sequenz dafür vor, die einfach in umgekehrter Reihenfolge abläuft (das kann man mit REVERSE FRAMES ΔBILDER UMKEHREN einfach erstellen), aber das erhöht natürlich auch die Dateigröße und den Zeitaufwand. Daher kann man sich eines Tricks bedienen. Er verlangsamt die Animation allerdings etwas.

Man erstellt eine Filmsequenz mit dem Instanznamen Controller mit drei Frames:

1. Frame

```
Stop();
```

2. Frame

```
TellTarget (rückwärts zu spielender Film oder Zeitleiste)
prevFrame ();
}
```

3. Frame

```
gotoAndPlay (2);
```

Dem Button, der das Rückwärtsspielen auslösen soll, gibt man nur den Befehl, die Filmsequenz zu starten. Wenn eine Zeitleiste bis zu einem bestimmten Punkt rückwärts gespielt werden soll, muss man den Loop nur in eine IF-Bedingung setzen, welche die Zahl der Rückschritte abpasst.

Wie lässt man ein Objekt eine Outline (z. B. die eines Buchstabens) verfolgen?

Viele spielen hierbei auf den Lensflare bei Flashworker an ...

Das ist kein Problem! Man zerlegt den Buchstaben in Zeichenwege und geht mit dem Tintenfass (nicht dem Fülleimer) auf den Pfad und setzt so eine Outline auf den Buchstaben! Diese kopiert man in den Zwischenspeicher und fügt zum Objekt eine Bewegungsführung hinzu! In diese neu entstandene Ebene kopiert man dann die Outline und kann an ihr eine Bewegungsführung ausrichten.

Wie importiert man Bilder mit transparenten Verläufen (Alpha)?

Dafür wendet man das PNG-Format. Wenn man z. B. in Photoshop einen Alphakanal erzeugt, um einen komplizierten Rahmen zu ziehen oder einen Verlauf in der Transparenz zu erstellen, kann man diesen Alphakanal per PNG in Flash übernehmen.

Kann man eine Maske mit Verlauf erstellen?

Nein! Aber wie so oft bei Flash gibt es Tricks! Wenn ein einfarbiger Hintergrund vorhanden ist, kann man einen Verlauf statt einer Maske verschieben., also einen Verlauf erstellen, der in der Mitte transparent ist. Wenn der Hintergrund nicht einfarbig ist, gibt es leider nicht viele Möglichkeiten. Unter Umständen ist es eine Lösung, ein PNG mit Verlauf zu verwenden!

Wie stellt man einen Scrollbalken her?

In Flash 2 wäre es einfach, ein Symbol über x Frames zu tweenen – von oben nach unten. Die Scrollbuttons machen dann nur noch `Goto next frame` und `Goto previews frame`.

Seit Flash 4 gibt es eine andere Möglichkeit. So könnte man z. B. eine Zeile weiter scrollen:

```
textfeld.scroll += 1;
```

Man kann auch eine exakte Zeile anspringen:

```
ausgabe.scroll = 30;
```

Wie animiert man Textfelder?

Die Font Outlines müssen mit eingebunden sein – dann funktioniert es. Wenn das Textfeld dann in einem Symbol ist, kann man es für alle Tweenings verwenden.

Wie verändert man den Buffer des Streaming Sounds?

Unter Umständen sind die fünf Sekunden Soundbuffer bei schnellen Animationen zu klein dimensioniert. Ab Flash 4 kann man diesen Wert ändern:

Mit `_soundbuftime` setzt man die Zeit in Sekunden für den Puffer des gestreamten Sounds für den gesamten Film fest.

`_soundbuftime = 10` bedeutet also, dass zehn Sekunden des Sounds vorgeladen werden, bevor der Film weiterspielt.

Wie erstellt und verwendet man Texturfüllungen?

Zuerst importiert man sich das Bild, mit dem man das Objekt füllen möchte. Dabei sollte man möglichst kleine Dateien benutzen. Die Grafik befindet sich nun auf der Bühne. Unter MODIFIZIEREN findet man die Funktion TEILEN (STRG + B). Das Bild ist nun zerlegt und kann mit der Pipette zu den Farben aufgenommen werden. Wenn man damit ein Objekt füllt, kachelt Flash das Bild als Textur.

Wie kann man ein geladenes SWF bzw. MovieClip positionieren?

Per `LoadMovie` geladene Filme werden immer an der linken oberen Ecke übereinander positioniert. Es gibt also zwei Möglichkeiten: Entweder macht man den geladenen Film gleich groß und positioniert so exakt. Oder man positioniert mit

```
loadMovie ("zweitermovie1.swf", _droptarget=1);
```

So wird genau an der Mitte der ersten Filmgröße ausgerichtet, und das funktioniert auch bei Fullscreen.

Eine andere Möglichkeit ist:

```
_level100._x = 100;
```

Die Zahl nach `_level` ist die Ebene, wohin das SWF geladen wurde.

Wie kann man mit Flash 4 runden?

```
   Set Variable: "INTzahl" = INT (zahl)
If (zahl - INTzahl >= 0.5)
   Set Variable: "INTzahl" = INTzahl + 1
Else
   If (zahl - INTzahl <= -0.5)
   Set Variable: "INTzahl" = INTzahl -1
End If
```

Was ist der Flash 4-Rotationsbug, und wie vermeide ich ihn?

Leider verkleinert Flash bei einer Rotation, deren Wert nicht ein Vielfaches von 0.25 ist (z. B. 1.25, 43.75 – 18.5), das gedrehte Objekt. Kommt der Rotationswert gar aus einer Berechnung, kann dieser etliche Stellen hinter dem Komma haben. Um dieses Problem zu umgehen, sollte nach jeder Rotation die Scalierung des Objektes wieder auf 100 % gesetzt werden.

Warum »klickt« mein Sound immer?

Womit wir gleich geklärt hätten, wo man noch Antworten findet:
http://www.macromedia.com/support/flash/ts/documents/soundclick.htm

Was Flasher noch brauchen

Fehler beim Öffnen der Datei

Wer hat das beim Doppelklick auf das FLA noch nicht gesehen? Es gibt einen kleinen Bug in der deutschen Übersetzung von Flash 5. Zwar fragt die deutsche Version nach dem Registry-Schlüssel »Flash.Film«, in die Registry HKEY_CLASSES_ROOT schreibt er es aber als »Flash.Movie«. Also einfach die Registry nach »Flash.Movie« durchsuchen und dies gegen »Flash.Film« tauschen. Es kann sein, dass dieser Fehler nur bei Updates der englischen Version auftritt.

Wo finde ich die aktuelle Bugliste?

Eine aktuelle Bugliste »Issues with the Macromedia Flash Player 5« finden Sie unter www.macromedia.com/support/flash/ts/documents/player5_issues.htm .

Wie sehe ich der SWF-Datei an, welche Version sie hat?

Man öffnet das SWF einfach in einem Hexeditor! Die ersten drei Bytes sind die Signaturen (F,W,S), danach kommt die Version. 05 wäre folglich dann Flash 5! Mehr dazu findet man bei www.openswf.org

Wie kann ich den Mediaplayer aus Flash steuern?

```
fscommand: "exec", chr(34) add "c:/windows/system/actmovie.exe"
 add chr(34)add chr(9) add chr(34) add"x:/Film.avi" add chr(34)
add chr(9) add"/fullscreen").
```

Wie verhindert man, dass ein externes Textfile (oder PLs, CGIs ...) gecached wird?

Um das Problem zu umgehen, muss der Aufruf des CGI-, PHP-, oder eines anderen Skripts mit einem weiteren, sich ändernden Parameter erfolgen.

Bsp:

```
"/cgi-bin/meinscript.pl?"&random(99999)
```

Gleiches gilt auch für TXT-Dateien oder andere Formate, die gelesen werden sollen. Das ist kein Bug von Macromedia, sondern ein so genanntes »Feature« von Internet Explorer. So umgeht man es.

Wie »rüttelt« man einen Browser?

Um einen Browser aus Flash quasi per Knopfdruck zu rütteln, braucht man nur einen simplen GetURL-Befehl:

```
Get URL ("JavaScript:{for (i = 25; i > 0; i--) {for (j = 3; j > 0;
j--){self.moveBy(0,i);self.moveBy(i,0);self.moveBy(0,-i);self.moveBy
(-i,0);}}} ", window="_self")
```

In HTML ist dafür kein Code nötig!
Rütteln in HTML funktioniert wie folgt:

```
<SCRIPT LANGUAGE="JavaScript">

<!--Begin
netscape = (navigator.appName == "Netscape");
n4 = netscape && (parseInt(navigator.appVersion) >= 4);
explorer = (navigator.appName == "Microsoft Internet Explorer");
ie4 = explorer && (parseInt(navigator.appVersion) >= 4);

function schuettel(n) {
if (n4 || ie4) {
for (i = 13; i > 0; i--) {
for (j = n; j > 0; j--) {
self.moveBy(0,i);
self.moveBy(i,0);
self.moveBy(0,-i);
self.moveBy(-i,0);
         }
      }
   }
}
// End -->
</script>
<A HREF=javascript:schuettel();> </a>
```

In Flash muss nun nur noch an der Stelle, wo es abgespielt werden soll, der Get URL-Befehl eingetragen werden: `javascript:move(n)`; Hierbei ist `n` dabei die Länge des Schüttelns.

Wie öffne ich einen Browser »Fullscreen«?

1. Diesen Code ins HTML-Dokument einfügen (nichts modifizieren):

```
<script language="JavaScript">
<!-- hide from JavaScript-challenged browsers

function openWindow() {
  popupWin = window.open('intro.swf', 'remote', 'toolbar=0,loca-
tion=0,
directories=0,menubar=0,scrollbars=0,resizeable=0,fullscreen=1')
}

// done hiding -->
</script>
```

2. Die Aktion, um das aufzurufen, wäre GetURL (`"openWindow()"`).

Wie kann ich Variablen aus Flash per JavaScript auslesen?

Diesen Code ins HTML-Dokument einfügen (nichts modifizieren):

```
<script language="JavaScript" fptype="dynamicanimation">
function f4_DoFSCommand(cmd, args)
{
  var InternetExplorer = navigator.appName.indexOf("Microsoft")
!= -1;
  var movieObj = InternetExplorer ? f4 : document.f4;

  if (cmd==getvars)
 {
    var JSVariable=movieObj.GetVariable("/target/:FlashVaria-
ble");
  }
</script>
```

Aus Flash heraus `FSCommand("getvars"," ")` aufrufen (Achtung, beim Export FSCommand aktivieren).
Aber es geht auch ohne FSCommand:

```
GetURL("javascript:{f4_DoFSCommand("getvars"," ")}")
```

Wie kann ich den Browser-Cache verhindern?

```
<META HTTP-EQUIV="Expires" CONTENT="Tue, 01 Jan 1980 1:00:00 GMT">
<meta http-equiv="Test" content="no-cache">
```

Das wird in den Head eingetragen, und der Browser dürfte die Seite nicht speichern!

Aber das bringt nur bei vereinzelten Browserversionen Erfolg. Die optimale, universelle Lösung gibt es nicht, aber Erfahrungen zeigen, dass es auch aus HTML heraus sinnvoll ist, eine URL mit einem Zufallsparameter aufzurufen. Das funktioniert aber leider nur per JavaScript.

Wie mache ich meinen Film transparent zum Hintergrund?

Der Hintergrund einer Flash-Seite kann transparent erstellt werden, so dass man die Hintergrundfarbe oder das Hintergrundbild der HTML-Seite hindurchscheinend sehen kann. Allerdings funktioniert das nur ab dem Internet Explorer 4.0 für Windows. Auch im Stand-alone-Projektor kann man mit Flash keinen transparenten Hintergrund erzeugen – dafür bräuchte man spezielle Tools. Es wird dann die eingestellte Hintergrundfarbe angezeigt.

Und so geht es: Unter Datei · Einstellungen für Veröffentlichungen den Reiter HTML wählen und hier unter Fenstermodus · Durchsichtig ohne Fenster einstellen. Der erforderliche Code wird dann beim Veröffentlichen in die HTML-Datei geschrieben. Man kann auch den HTML-Quellcode direkt bearbeiten. Unter OBJECT muss dann stehen `<param name=wmode value=transparent>` und bei EMBED `wmode=transparent`.

Wie lasse ich einen Projektor von CD automatisch starten (autorun)?

Die Datei autorun.inf muss im Root der CD stehen. Es ist ein normales Textfile:

```
[autorun] open=projektor.exe (der Aufruf)
shell\install=kontextmen&üeintrag
```

Das Ü ist ein Shortcut und wird unterstrichen!

```
shell\install\command=projektor.exe
```

Hier steht das Gleiche wie bei »open«. Es handelt sich dabei um das Kommando, das beim Kontextmenüeintrag ausgeführt wird.

```
icon=projektor.ico
```

Wenn man ein eigenes Icon will, muss es auch im Root der CD sein.

Mehr dazu unter http://www.wolfgang-wirth.de/prod/arun/autorun.htm.

Wie kann man MIDI aus Flash abspielen?

Aus Flash leider gar nicht ... wie so oft in Flash gibt es aber Tricks:

Man erstellt einfach ein HTML-File, welches das MIDI-File lädt! Per GetURL kann man dieses dann starten oder stoppen (also HTML mit MIDI laden oder ohne). Wenn dafür kein extra Browserfenster aufgehen soll, empfiehlt es sich, ein Miniframe (ein Pixel breit vielleicht) einzurichten!

Worauf muss man achten, wenn man mit Flash für TV produziert?

Bei 16:9-Format legt man sein Projekt auf 1024 x 576 an. Beim Ausspielen wird das Ganze auf dem Band auf 720 x 576 wieder gestaucht, da sonst das bekannte »Eierkopfproblem« auftritt. Die Pixel am Computermonitor sehen aus wie kleine Quadrate. Am Fernseher sind es eher längliche Stäbchen. Wenn man also einen Kreis sozusagen vom Computermonitor zum Fernseher eins zu eins transferiert, würden sich die Proportionen des Kreises verzerren, da aus den quadratischen Pixeln nun längliche geworden sind. Also stauchen wir, um die Proportionen zu erhalten.

Die normale Auflösung für den Fernseher ist, solange man am Monitor arbeitet, 768 x 576. Beim Ausspielen wird das Ganze wiederum auf 720 x 576 gestaucht. Umgedreht funktioniert es genauso: Fernsehmaterial mit 720 x 576 holen und dann auf 768 x 576 stretchen, damit ein Kreis ein Kreis bleibt.

Woher bekommt man alte Plug-ins zum testen?

http://www.macromedia.com/support/flash/ts/documents/oldplayers.htm

Vorsicht – derzeit haben alle Player die alte Dateigröße! Der Verdacht liegt nahe, dass es sich versehentlich um aktuelle Player handelt. Bis zum Erscheinen des Buches kann dieser Fehler behoben sein.

Wishlist Flash 6

Was Flashern den Alltag erleichtern würde ...

Der nahezu ungefilterte Stand der Wünsche von macromedia.general.germany am 15. November 2000.

- Schriftunterstützung von Zeichensätzen über ASCII 255, um ttf-interne Fremdsprachenzeichen verwenden zu können
- Bei allen Fehlermeldungen im Output soll dabei stehen, welches SWF die Fehlermeldung verursacht und in welcher Zeile es einen Fehler gibt (bei loadmovie-Fehlern z. B.)
- Die HTML-Textfelder sollten Tabellen unterstützen
- Es fehlt die Möglichkeit, nicht nur die Eigenschaften von Movies dynamisch zu setzen (wie bei `_x, _height` etc.), sondern auch bei Eingabeelementen wie: Farbe, Schriftart, Größe von Texten, Setzen der Höhe von Textboxen (wenn man die Weite von Movies setzt, dann klappt das, und das Eingabeelement wird breiter. Verändert man hingegen die Höhe, wird das Ganze verzerrt)
- Es fehlt ganz dringend ein flush-Befehl, der Veränderungen sofort darstellt, nicht erst nach dem Ende des Funktionsaufrufes
- Die Funktion `md5` fehlt völlig
- Abfragen der Breite eines dynamischen Textfeldes
- Breite eines dynamischen Textfeldes automatisch an die Länge des Inhalts anpassen
- Funktion `toGMTString` wie in JavaScript
- Dynamisches Ersetzen von Bildern! (Wird wohl kaum kommen, denn dann wäre Generator wieder ein Stück mehr ersetzbar.)
- Dynamisches Ändern der Moviegröße
- Eine z-Achse
- Unterstützung von gängigen 3D-Schnittstellen (direct3d, openGL und glide)
- Elemente zur Beleuchtung von Objekten
- Schnellere Darstellung von Animationen; Flash geht sehr schnell in die Knie, da der Prozessor alles erledigen muss. Die Grafikkarte schläft in der Zwischenzeit ein!

- Man kann noch immer keine großen Objekte in Flash bearbeiten ... auch wenn man auf Ansicht 25 % schaltet, ist nicht alles sichtbar. Zurzeit muss man tricksen, indem man alle Inhalte auf 10 % reduziert, in der 800 %-Ansicht alles bearbeitet und schließlich wieder alles auf 100 % skaliert – das ist zu umständlich!
- Bitmaps mit den Ebenenattributen von Photoshop versehen können
- ActionScript-Editor mit verbesserter Druckoption
- ActionScript-Editor, der eigene Schriftarten anzeigt; man kann kaum ein großes W von einem kleinen unterscheiden
- Vereinfachung von Projektabwicklungen durch Versionskontrolltools und verbesserten Debug-Methoden
- Eine simple Preloader-Funktion
- AutoCode wie bei Visual Basic
- Verschlüsselung von Daten (xml)
- Multithreading (mehrere Funktionen zur gleichen Zeit ausführen)
- ActionScript-Referenz mit Beispielen!
- Kein neues Plug-in *g*.

Eigene Wünsche kann man in englischer Sprache an flash-wish@marcomedia.com senden.

Compiler-Fehlermeldungen

Es folgt eine Liste von Fehlermeldungen, die vom Flash Compiler ausgegeben werden. Zu jeder Meldung wird eine Erklärung gegeben, damit Sie die Probleme in Ihren Filmdateien leichter beheben können.

▸ Gerald Marischka

Dieser Abschnitt wurde der Hilfedatei von Flash entnommen.

Fehlermeldung	Beschreibung
Eigenschaft `<property>` existiert nicht	Es wurde eine nicht vorhandene Eigenschaft gefunden. Beispiel: `x = _green` ist ungültig, weil die Eigenschaft `_green` nicht vorhanden ist.
Nach Operator `<operator>` muss ein Operand folgen.	Es wurde ein Operator ohne Operand gefunden. Beispiel: In x = 1 + ist nach dem Operator + ein Operand erforderlich. Nach einem Operator steht ein ungültiger Operand. Beispiel: `trace(1+);` ist syntaktisch fehlerhaft.
Syntaxfehler	Diese Meldung wird ausgegeben, wenn ein nicht weiter spezifierter Syntaxfehler gefunden wird.
Feldname hinter '.'-Operator erwartet.	Sie müssen einen gültigen Feldnamen eingeben, wenn Sie die Syntax objekt.feld verwenden.

Expected `<token>`	Ein ungültiges oder nicht erwartetes Token wurde gefunden. Beispiel: In der folgenden Syntax ist das Token `foo` ungültig. Es wird das Token `while` erwartet.
	`do {`
	`trace (i)`
	`} foo (i < 100)`

Initialisierliste muss mit `<terminator>` abgeschlossen werden.	In einer Liste für Objekt- oder Arrayinitialisierliste fehlt der Abschluss durch] oder }.
Bezeichner erwartet	An Stelle eines Bezeichners wurde ein unerwartetes Token gefunden. Im folgenden Beispiel ist 3 kein gültiger Bezeichner. `var 3 = 4;`
Das JavaScript-Konstrukt <construct> wird nicht unterstützt.	Es wurde ein JavaScript-Konstrukt gefunden, das von ActionScript nicht unterstützt wird. Diese Nachricht wird angezeigt, wenn folgende JavaScript-Konstrukte verwendet werden: `void, switch, try, catch` oder `throw`.

Fehlermeldung	Beschreibung
Die linke Seite eines Zuweisungsoperators muss eine Variable oder eine Eigenschaft sein.	Es wurde ein Zuweisungsoperator gefunden. Auf der linken Seite der Zuweisung steht jedoch weder eine gültige Variable noch eine Eigenschaft.
Anweisungsblock muss mit '}' abgeschlossen werden.	Innerhalb von geschweiften Klammern wurde eine Gruppe von Anweisungen deklariert, aber die abschließende Klammer fehlt.
Ereignis erwartet	Es wurde ein Handler On(MouseEvent) oder onClipEvent deklariert, aber kein Ereignis angegeben, oder an der für das Ereignis vorgesehenen Stelle wurde ein unerwartetes Token gefunden.
Nicht zulässiges Ereignis	Das Skript enthält ein ungültiges Maus- oder Sequenzereignis. Eine Liste der gültigen Maus- und Sequenzereignisse finden Sie unter den Flash-Hilfeeinträgen »On(MouseEvent)« und »OnClipEvent« im Flash-Hilfe-Kapitel »Referenz zu ActionScript«.
Tastencode erwartet	Sie müssen einen Tastencode angeben. Eine Liste der Tastencodes finden Sie in der Flash-Hilfe.
Nicht zulässiger Tastencode	Der angegebene Tastencode existiert nicht.
Nicht notwendige Daten als Anhang gefunden	Das Skript oder der Ausdruck wurde ordnungsgemäß ausgewertet. Die darauf folgenden Zeichen konnten jedoch nicht geparst werden.
Unzulässige Funktion	Eine Deklaration einer benannten Funktion wurde als Ausdruck verwendet. Deklarationen von benannten Funktionen müssen Anweisungen sein. Gültig: `function sqr (x) { return x * x; }` Ungültig: `var v = function sqr (x) { return x * x; }`
Funktionsname erwartet	Der für diese Funktion angegebene Name ist kein gültiger Funktionsname.
Parametername erwartet	In einer Funktionsdeklaration wurde ein Name für einen Parameter (Argument) erwartet. Es wurde aber ein unerwartetes Token gefunden.
'else' ohne entsprechendes 'if' festgestellt	Es wurde die Anweisung else gefunden. Dieser geht jedoch kein if voran. Sie können else nur in Verbindung mit einer if-Anweisung verwenden.
Szenenname muss eine Zeichenfolge in Anführungszeichen sein	Das Szenenargument einer Aktion `gotoAndPlay`, `gotoAndStop` oder `ifFrameLoaded` ist vom falschen Typ. Das Szenenargument muss eine Zeichenfolgenkonstante sein.
Interner Fehler	Im ActionScript-Compiler ist ein interner Fehler aufgetreten. Senden Sie bitte die FLA-Datei, bei der dieser Fehler aufgetreten ist, und detaillierte Anweisungen, wie die Meldung reproduziert werden kann, an Macromedia.
Hexadezimale Zeichen nach 0x erwartet	Es wurde die Zeichenfolge 0x gefunden. Auf diese Zeichenfolge folgt jedoch keine gültige Hexadezimalzahl.

Fehlermeldung	Beschreibung
Fehler beim Öffnen von include-Datei [#include file]	Beim Öffnen einer mit der Direktive include eingebundenen Datei ist ein Fehler aufgetreten. Dieser Fehler tritt z. B. auf, wenn die Datei nicht vorhanden ist oder ein Festplattenfehler vorliegt.
Ungültige #include-Anweisung	Eine include-Direktive wurde fehlerhaft geschrieben. Eine include-Direktive muss die folgende Syntax aufweisen: #include »einedatei.as«
Mehrzeiliger Kommentar wurde nicht abgeschlossen	Bei einem mehrzeiligen Kommentar, der mit /* beginnt, fehlen die Zeichen */ als Abschluss.
Zeichenfolgenliteral wurde nicht ordnungsgemäß abgeschlossen	Bei einem Zeichenfolgenliteral, das mit einem einfachen oder doppelten Anführungszeichen beginnt, fehlt das schließende Anführungszeichen.
Funktion `<function>` hat `<count>` Parameter	Eine Funktion wurde aufgerufen. Dabei wurde eine unerwartete Anzahl an Parametern gefunden.
Eigenschaftenname in `GetProperty` erwartet	Die Funktion `getProperty` wurde aufgerufen. Beim zweiten Argument handelt es sich jedoch nicht um den Namen einer Filmsequenzeigenschaft.
Parameter `<parameter>` darf nicht mehrfach deklariert werden.	In der Parameterliste einer Funktionsdeklaration ist ein Parametername mehrmals aufgeführt. Alle Parameternamen müssen eindeutig sein.
Variable `<variable>` darf nicht mehrfach deklariert werden.	In einer var-Anweisung ist ein Variablenname mehrfach aufgeführt. In einer einzelnen var-Anweisung müssen alle Variablennamen eindeutig sein.
`'on'`-Handler dürfen nicht in andere 'on' verschachtelt werden.	Ein on-Handler wurde innerhalb eines anderen on-Handlers deklariert. Alle on-Handler müssen sich auf der obersten Ebene einer Aktionsliste befinden.
Anweisung muss innerhalb eines `[on]`-Handlers stehen.	In den Aktionen für eine Schaltflächeninstanz wurde eine Anweisung ohne einen umgebenden on-Block deklariert. Alle Aktionen für eine Schaltflächeninstanz müssen sich in einem on-Block befinden.
Anweisung muss innerhalb eines `onClipEvent`-Handlers stehen.	In den Aktionen für eine Filmsequenzinstanz wurde eine Anweisung ohne einen umgebenden `onClipEvent`-Block deklariert. Alle Aktionen für eine Filmsequenzinstanz müssen sich in einem `onClipEvent`-Block befinden.
Mausereignisse sind nur bei Schaltflächeninstanzen zulässig.	Ein Handler für Schaltflächenereignisse wurde in einer Bildaktionsliste oder in einer Aktionsliste einer Filmsequenzinstanz deklariert. Schaltflächenereignisse sind nur in Aktionslisten von Schaltflächeninstanzen zulässig.
Sequenzereignisse sind nur bei Filmsequenzinstanzen zulässig.	Ein Handler für Sequenzereignisse wurde in einer Bildaktionsliste oder in einer Aktionsliste einer Schaltflächeninstanz deklariert. Sequenzereignisse sind nur in den Aktionslisten von Filmsequenzinstanzen zulässig.

Aktion trace

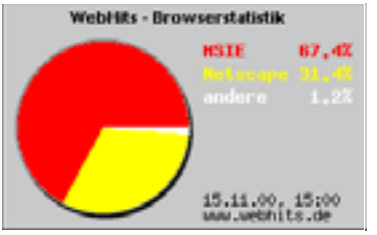
Mit Hilfe der Aktion `trace` in einem Skript können Sie Informationen an das Ausgabefenster senden. Sie können so, wenn Sie beispielsweise einen Film oder eine Szene testen, bestimmte Programmierhinweise an das Fenster senden.
Sie können auch festlegen, dass beim Drücken einer Schaltfläche oder Abspielen eines Bildes bestimmte Ergebnisse angezeigt werden. Die Aktion `trace` ist mit der Anweisung `alert` in JavaScript vergleichbar.

Wenn Sie die Aktion `trace` in einem Skript einsetzen, können Sie Ausdrücke als Argumente verwenden. Im Filmtestmodus wird der Wert eines Ausdrucks im Ausgabefenster angezeigt, wie im folgenden Beispiel dargestellt:

```
onClipEvent(enterFrame){
   trace("onClipEvent enterFrame" + enterFrame++)
}
```

Webstatistiken

Abbildung 1 ▸
Browser



▸
Gerald Marischka

Durch die freundliche Erlaubnis von Peter Decker von www.webhits.de darf ich Ihnen die aktuellen Webstatistiken vom 15.11.2000 präsentieren.

Browser

Browser- und Systemübersicht Quelle: www.webhits.de

Browser	
Microsoft Internet Explorer	67.4 %
Netscape	31.4 %
Unbekannt	0.6 %
WebWasher	0.4 %
Opera	0.1 %
IBrowse	0.1 %
AWeb	0.0 %
Voyager	0.0 %
Browserdetails (I)	
MS IE 5	60.9 %
Netscape 4	30.5 %
MS IE 4	6.2 %
unbekannt	0.6 %
Netscape 3	0.6 %
MSIE 3	0.3 %

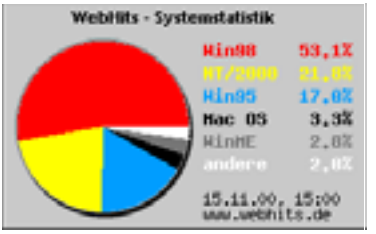
Browser- und Systemübersicht Quelle: www.webhits.de

WebWasher IE 5	0.2 %
Netscape 6	0.2 %
WebWasher NS 4	0.1 %
Opera	0.1 %
IBrowse	0.1 %
AWeb	0.0 %
Voyager	0.0 %
andere	0.2 %
Browserdetails II	
MS IE 5.0	29.6 %
MS IE 5.5	17.1 %
MS IE 5.01	14.1 %
Netscape 4.7	9.3 %
MS IE 4.01	5.6 %
Netscape 4.5	4.4 %
Netscape 4.51	3.1 %
Netscape 4.6	2.8 %
Netscape 4.73	2.8 %
Netscape 4.75	1.7 %
Netscape 4.05	1.5 %
Netscape 4.08	0.9 %
Netscape 4.72	0.8 %
unbekannt	0.6 %
Netscape 4.07	0.6 %
Netscape 4.61	0.5 %
Netscape 3.01	0.5 %
Netscape 4.03	0.5 %
Netscape 4.01	0.4 %
MS IE 4.0	0.3 %
Netscape 4.06	0.3 %
Netscape 4.04	0.3 %
MS IE 4.5	0.3 %
Netscape 4.76	0.2 %
Netscape 4.74	0.2 %
Netscape 6.0	0.2 %
MSIE 3.02	0.2 %
Opera	0.1 %
WebWasher IE 5.01	0.1 %

Browser- und Systemübersicht Quelle: www.webhits.de

Netscape 4.71	0.1 %
MS IE 3.0	0.1 %
IBrowse	0.1 %
WebWasher IE 5.5	0.1 %
Netscape 3.04	0.0 %
WebWasher NS 4.7	0.0 %
AWeb	0.0 %
WebWasher IE 5.0	0.0 %
Voyager	0.0 %
MS IE 5.0b1	0.0 %
MS IE 3.01	0.0 %
Netscape 4.5b1	0.0 %
Andere	0.6 %
Systemdetails	
Win 98	53.1 %
NT/2000	21.8 %
Win 95	17.0 %
Mac OS	3.3 %
Win ME	2.8 %
X11	0.8 %
Win 3.x	0.3 %
AmigaOS	0.2 %
OS/2	0.1 %
andere/unbekannt	0.6 %
Bildschirmauflösungen	
1024 x 768	54.6 %
800 x 600	29.7 %
1152 x 864	6.6 %
1280 x 1024	5.8 %
1600 x 1200	1.2 %
640 x 480	0.9 %
andere	1.2 %
Farbtiefen	
16 Bit	48.4 %
32 Bit	35.8 %
24 Bit	13.3 %
8 Bit	1.6 %
18 Bit	0.8 %

Abbildung 2 ▶
Systemstatistik

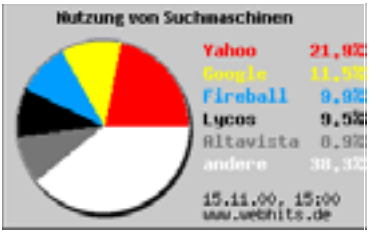


Browser- und Systemübersicht Quelle: www.webhits.de

JavaScript	
JavaScript 1.3	54.3 %
JavaScript 1.5	39.9 %
JavaScript 1.2	4.2 %
JavaScript 1.4	0.5 %
JavaScript 1.1	0.2 %
JavaScript 1.0	0.1 %
nicht aktiviert	0.8 %
Java	
aktiviert	98.5 %
nicht aktiviert	1.5 %
CSS	
aktiviert	99.3 %
nicht aktiviert	0.7 %
Cookies	
aktiviert	97.4 %
nicht aktiviert	2.6 %

Werte für Auflösung, Farbtiefe und CSS auf Basis von 11.900 Anwendern mit JavaScript 1.2-fähigen Browsern. Werte für Java basieren auf 11.100 untersuchten Browsern mit aktiviertem JavaScript.

Abbildung 3 ▶
Nutzung von Suchmaschinen



Installierte Plug-ins

Netscape Default Plug-in	96.1 %
LiveAudio	90.2 %
Shockwave Flash	87.7 %
NPAVI32 Dynamic Link Library	79.5 %
Headspace Beatnik Player Stub V1.0.0.1	69.4 %
RealPlayer(tm) G2 LiveConnect-Enabled Plug-in (32 Bit)	63.2 %
Adobe Acrobat	55.7 %
QuickTime Plug-in	48.8 %
Shockwave for Director	45.0 %
Microsoft® Windows Media Services	29.7 %
Netscape Media Player	21.5 %
Windows Media Player Plug-in Dynamic Link Library	19.6 %
Netscape vCalendar Plug-in	16.6 %
QuickTime Plug-in 4.0.1	12.9 %
MJuice Download Plug-in	10.5 %
FUN KIT Plug-in Dynamic Link Library (1.506)	10.4 %
CorporateTime vCalendar Plug-in	10.2 %

Verteilung unter 3100 Netscape-Browsern, nur Plug-ins mit mehr als 10 % Verbreitung.

Nutzung von Suchmaschinen

Yahoo	21.9 %
Google	11.5 %
fireball.de	9.9 %
Lycos	9.5 %
Altavista	8.9 %
web.de	7.9 %
Infoseek	5.1 %
Meta.Ger	5.0 %
MSN Websuche	4.7 %
Dino	2.5 %
Excite	2.3 %
AllesKlar	1.6 %
Bellnet	0.9 %
AV Webverzeichnis	0.8 %
alltheweb.com	0.6 %
acoon.de	0.6 %
Abacho	0.6 %
search.netscape.com	0.5 %
excite.com	0.5 %
Metacrawler	0.5 %
AOL Search	0.5 %
Hotbot	0.4 %
sharelook.de	0.4 %
AOL Netfind	0.4 %
Apollo7	0.3 %
search.ch	0.3 %
Looksmart	0.2 %
goto.com	0.2 %
Infoseek GO	0.2 %
Austronaut	0.2 %
Dogpile	0.1 %
crawler.de	0.1 %
Euroseek	0.1 %
directory.netscape.com	0.1 %
XFind	0.1 %
northernlight.com	0.1 %

Verteilung unter 14.900 verwendeten Suchergebnissen.

HTML 3.2-ASCII-Zeichen

Auf mehrfachen Wunsch möchte ich Ihnen hier den »guten alten« ASCII-Zeichensatz in einer Tabelle zusammenfassen.

▸
Gerald Marischka

ASCII-Wert	Zeichen	Name	Erklärung
#009			Tabulator
#010			Zeilenvorschub
#013			Eingabetaste
#032			Leertaste
#033	!		Ausrufezeichen
#034	"	"	Anführungszeichen
#035	#		Doppelkreuz
#036	$		Dollar
#037	%		Prozent
#038	&	&	Undzeichen
#039	'		Apostroph
#040	(		runde öffnende Klammer
#041	)		runde schließende Klammer
#042	*		Stern
#043	+		Plus
#044	,		Komma
#045	-		Minus
#046	.		Punkt
#047	/		Schrägstrich
#048	0		Ziffer 0
#049	1		Ziffer 1
#050	2		Ziffer 2
#051	3		Ziffer 3
#052	4		Ziffer 4
#053	5		Ziffer 5
#054	6		Ziffer 6
#055	7		Ziffer 7
#056	8		Ziffer 8
#057	9		Ziffer 9

ASCII-Wert	Zeichen	Name	Erklärung
#058	:		Doppelpunkt
#059	;		Semikolon
#060	<		Kleiner
#061	=		Gleich
#062	>		Größer
#063	?		Fragezeichen
#064	@		Add-Zeichen
#065	A		
#066	B		
#056	C		
#068	D		
#069	E		
#070	F		
#071	G		
#072	H		
#073	I		
#074	J		
#075	K		
#076	L		
#077	M		
#078	N		
#079	O		
#080	P		
#081	Q		
#082	R		
#083	S		
#084	T		
#085	U		
#086	V		
#087	W		
#088	X		
#089	Y		
#090	Z		
#091	[		Eckige Klammer auf
#092	\		Backslash
#093	]		Eckige Klammer zu
#094	^		Potenz
#095	_		Unterstrich
#096	`		Akzent

ASCII-Wert	Zeichen	Name	Erklärung
#097	a		
#098	b		
#099	c		
#100	d		
#101	e		
#102	f		
#103	g		
#104	h		
#105	i		
#106	j		
#107	k		
#108	l		
#109	m		
#110	n		
#111	o		
#112	p		
#113	q		
#114	r		
#115	s		
#116	t		
#117	u		
#118	v		
#119	w		
#120	x		
#121	y		
#122	z		
#123	{		Geschweifte Klammer auf
#124	|		Senkrechter Strich
#125	}		Geschweifte Klammer zu
#126	~		Tilde
#127			
#128	€		Eurozeichen
#129			
#130	‚		Komma
#131	*ƒ*		Florin
#132	»		Anführung rechts
#133	…		Ellipse
#134	†		Dagger
#135	‡		Doppel-Dagger

ASCII-Wert	Zeichen	Name	Erklärung
#136	ˆ		Zirkumflex
#137	‰		Promille
#138	Š		
#139	‹		Kleiner
#140	Œ		Großes OE
#141			
#142	Ž		
#143			
#144			
#145	‘		Einz. Anführung links
#146	’		Einz. Anführung rechts
#147	»		Anführung links
#148	«		Anführung rechts
#149	•		Bullet
#150	–		Em-Gedankenstrich
#151	—		En-Gedankenstrich
#152	˜		Tilde oben
#153	™		Trademark
#154	š		
#155	›		Größer
#156	œ		Kleines OE
#157			
#158	ž		
#159	Ÿ		Großer Y-Umlaut
#160			Spezielles Leerzeichen
#161	¡		Umgekehrtes Ausrufezeichen
#162	¢	¢	Cent
#163	£	£	Pfund
#164	¤	¤	Währung
#165	¥	¥	Yen
#166	¦	¦	Unterbrochener senkrechter Strich
#167	§	§	Paragraf
#168	¨	¨	Umlautpunkte
#169	©	©	Copyright
#170	ª	ª	Weibliches Ordinal
#171	«	«	Franz. Anführung links
#172	¬	¬	Nicht
#173	-		Leichter Gedankenstrich

ASCII-Wert	Zeichen	Name	Erklärung
#174	®	®	Registriertes Trademark
#175	¯	¯	Makrone
#176	°	°	Grad
#177	±	±	Plusminus
#178	²	²	Hochgestellte 2
#179	³	³	Hochgestellte 3
#180	´	´	Akzent
#181	µ	µ	My
#182	¶	¶	Paragraf
#183	·	·	Mittlerer Punkt
#184	¸	¸	Cedille
#185	¹	¹	Hochgestellte 1
#186	º	º	Männliches Ordinal
#187	»	»	Franz. Anführung rechts
#188	¼	¼	Bruch 1/4
#189	½	½	Bruch 1/2
#190	¾	&frac34	Bruch 3/4
#191	¿	¿	umgekehrtes Fragezeichen
#192	À	À	Großes A mit schwerem Akzent
#193	Á	Á	Großes A mit leichtem Akzent
#194	Â	Â	Großes A mit Zirkumflex
#195	Ã	Ã	Großes A mit Tilde
#196	Ä	Ä	Großer A-Umlaut
#197	Å	Å	Großes A mit Ring
#198	Æ	Æ	Großes AE
#199	Ç	Ç	Großes C-Cedille
#200	È	È	Großes E mit schwerem Akzent
#201	É	É	Großes E mit leichtem Akzent
#202	Ê	Ê	Großes E mit Zirkumflex
#203	Ë	Ë	großer E-Umlaut
#204	Ì	Ì	Großes I mit schwerem Akzent
#205	Í	Í	Großes I mit leichtem Akzent
#206	Î	Î	Großes I mit Zirkumflex
#207	Ï	Ï	großer I-Umlaut
#208	Ð	Ð	Großes Eth
#209	Ñ	Ñ	Großes N mit Tilde
#210	Ò	Ò	Großes O mit schwerem Akzent
#211	Ó	Ó	Großes O mit leichtem Akzent
#212	Ô	Ô	Großes O mit Zirkumflex

ASCII-Wert	Zeichen	Name	Erklärung
#213	Õ	Õ	Großes O mit Tilde
#214	Ö	Ö	Großer O-Umlaut
#215	×	×	Multiplizierzeichen
#216	Ø	Ø	Großes O mit Strich
#217	Ù	Ù	Großes U mit schwerem Akzent
#218	Ú	Ú	Großes U mit leichtem Akzent
#219	Û	Û	Großes U mit Zirkumflex
#220	Ü	Ü	Großer U-Umlaut
#221	Ý	Ý	Großes Y mit Akzent
#222	Þ	Þ	Großes Thorn
#223	ß	ß	Scharfes ß
#224	à	à	Kleines A mit schwerem Akzent
#225	á	á	Kleines A mit leichtem Akzent
#226	â	â	Kleines A mit Zirkumflex
#227	ã	ã	Kleines A mit Tilde
#228	ä	ä	Kleiner A-Umlaut
#229	å	å	Kleines A mit Ring
#230	æ	æ	Kleines AE
#231	ç	ç	Kleines C-Cedille
#232	è	è	Kleines E mit schwerem Akzent
#233	é	é	Kleines E mit leichtem Akzent
#234	ê	ê	Kleines E mit Zirkumflex
#235	ë	ë	Kleiner E-Umlaut
#236	ì	ì	Kleines I mit schwerem Akzent
#237	í	í	Kleines I mit leichtem Akzent
#238	î	î	Kleines I mit Zirkumflex
#239	ï	ï	Kleines I-Umlaut
#240	ð	ð	Kleines ETH
#241	ñ	ñ	Kleines N mit Tilde
#242	ò	ò	Kleines O mit schwerem Akzent
#243	ó	ó	Kleines O mit leichtem Akzent
#244	ô	ô	Kleines O mit Zirkumflex
#245	õ	õ	Kleines O mit Tilde
#246	ö	ö	Kleiner O-Umlaut
#247	÷	÷	Divisionszeichen
#248	ø	ø	Kleines O mit Strich
#249	ù	ù	Kleines U mit schwerem Akzent
#250	ú	ú	Kleines U mit leichtem Akzent
#251	û	û	Kleines U mit Zirkumflex

ASCII-Wert	Zeichen	Name	Erklärung
#252	ü	ü	Kleiner U-Umlaut
#253	ý	ý	Kleines Y mit Akzent
#254	þ	þ	Kleines Thorn
#255	ÿ	ÿ	Kleiner Y-Umlaut

Tastatureingaben und Tastencodewerte

In der Flash-Hilfe werden alle Tasten auf einer Standardtastatur und die entsprechenden Tastencodewerte angeführt, mit denen diese Tasten in ActionScript identifiziert werden können. Weitere Informationen finden Sie in der Beschreibung des Key-Objektes in der Referenz zu ActionScript in der Flash-Hilfe.

Die Autoren dieses Buches

Die Autoren des Buches wurden nach ihren Spezialgebieten ausgesucht. Sie sind Flashworker, Freelancer, Autoren anderer Bücher oder andere Koryphäen der Szene. Wer welches Kapitel geschrieben hat, haben wir jeweils beim Kapitel vermerkt. Durch die Angabe der E-Mail-Adressen möchten wir Ihnen die Möglichkeit geben, die Spezialisten selbst zu kontaktieren.

Autor	Kontaktadresse	Weitere Infos
Alexander Kutz	a.kutz@medienmeile.de www.medienmeile.de	Webdesign, Flash, Schulung Webworker seit 1996.
Boblgum	flashfreak@mysterion.de www.mysterion.de	Die »mysteriON group« ist ein Team von Webentwicklern, die sich mit so ziemlich allen Fragen bezüglich WWW befassen.
Belinda Baumgartner	belindab@aon.at www.belindab.com	Belinda Baumgartner arbeitet für Algo Software (www.algo.cc).
Björn Gromoll	bg@powerflasher.de www.powerflasher.de	
Carlo Blatz	cb@powerflasher.de www.powerflasher.de	
Christoph Aigner	christoph@alaris.at www.alaris.at	CTO und Mitbegründer der Alaris Informationsmanagement GmbH
Daniel Schulten	schulten@netzkern.com www.netzkern.com	Daniel Schulten von der Internet- und Werbeagentur netzkern beschäftigt sich seit drei Jahren mit Flash. Besonders die Verbindung von Flash mit Datenbanken und interaktiven Inhalten, die der Kunde selber pflegen kann, haben es ihm und seiner Agentur angetan.
Danny G. Franzreb	danny@franzreb.com www.franzreb.com	Danny ist der Kopf von franzreb.com, einer Webdesign-Agentur, die sich auf Flash spezialisiert hat. Er ist der Initiator von motioninmotion.com, einem progressiven Projekt, das von Flashdesignern aus aller Welt getragen wird.
Detlef Randerath	d.randerath@europop.net www.detlef.de	Detlef Randerath leitet die Multimedia-Division der EUROPOP AG in Düsseldorf. Er ist seit mehr als einem Jahrzehnt im Bereich Multimedia-Entwicklung tätig und wirkte an vielen nationalen und internationalen Projekten in leitender Position mit.
Dirk Schmeckthal	pieflash2be.com www.flash2be.com	

Autor	Kontaktadresse	Weitere Infos
Dominic Schech	dominik@schech.net www.schech.net	
Emrah Hircin	emrah@hircin.de www.emrahnet.de	Emrah Hircin, Webdeveloper aus Köln, ist Spezialist im Umgang mit Dreamweaver und Fireworks.
Eric Singhartinger	eric@esreverse.com www.esreverse.com	Eric Singhartinger ist Freelancer im Bereich Konzeption und Kreation. Er realisiert Sites aller Art für Agenturen auf der ganzen Welt. Jede seiner Sites ist innovativ und intuitiv in der Benutzerführung. Mit Flash hat er im letzten Jahr mehr als ein Dutzend dieser Sites realisiert.
Gerald Marischka	office@profipage.at www.profipage.at	
	grether@cyberpark.de	
Geza Varkuti	geza@esoundcity.com www.esoundcity.com	Geza Varkuti hat einen Abschluß am Konservatorium von Budapest, Jazz-Fakultät. Danach Produzent bei zahlreichen Schallplatten. Derzeit spezialisiert auf digitale Soundbearbeitung und Produktion.
Hassan Beigi	hb@powerflasher.de www.powerflasher.de	
Jan Schlünzen	jan@jobbernowl.de www.jobbernowl.de	
Jan C. Frischmut	Liquid@StarSites.de www.powerflasher.de	
Johannes Pohl	JohannesP@2-serious.de www.2-serious.de	
Joscha Feth	JoschaFeth@gmx.de www.flashtool.de	
Kai Schmeling	kschmeling@kaiowana.de www.kaiowana.de	Kai Schmeling arbeitet in der Agentur Animax - the e-community www.animax-e.com. Für Fragen zu Flash und Sound ist er der erste Ansprechpartner, da er der Krachmacher in der Agentur ist :-)
Marco Asbach	partylogger@partylogger.de www.partylogger.de	Marco Asbach ist Webmaster von PartyLoggers Universe.
Mario Mies	mario.mies@gmx.de www.flashbeat.de	
Markus Gotttscheber	mg@bo-ss.at www.bo-ss.at	
Martin Fleck	info@action-script.com www.action-script.com	Martin Fleck ist als Flash Developer besonders spezialisiert auf die Arbeit mit ActionScript.
Martin Plücker	mpluecker@diconnects.de www.diconnects.de	Martin Plücker von der DiCONNECTS GmbH ist spezialisiert auf Design und Programmierung im Internet mit HTML, Flash und Perl.

Autor	Kontaktadresse	Weitere Infos
Michael Bundscherer	michi@typolis.de www.notblue.de	Michi Bundscherer erstellte bereits Anfang 1997 »Typolis« – eine Seite über Typografie, Schrift und Gestaltung – die sich mittlerweile zu einer der beliebtesten Seiten für Typobegeisterte gemausert hat. Dieses Thema hat ihn seit damals nicht mehr losgelassen und so untersucht er heute als Mitgesellschafter von »notblue« unter anderem, wie Typografie die Benutzung der neuen Medien vereinfachen bzw. verschönern kann.
Michael Schinkel	mSchinkel@vision4net.de www.vision4net.de	Michael Schinkel von der Agentur vision4net ist spezialisiert auf Macromedia Flash und dynamische Webinhalte.
Peter Karsten	home@peter-karsten.de www.peter-karsten.de	Peter Karsten ist sorgt als Mitarbeiter von TELDO Global Communication mit seiner Internetprogrammierung für ein ständig wachsendes dynamisches Internet.
Peter Krempl	info@firstflash.de www.flashspiele.de	
Roland Grela	webmaster@flash-your-life.de www.flash-your-life.de	Roland Grela nutzt seit einem Jahr jede freie Minute, um sich mit Flash zu beschäftigen. Besonders ActionScripts haben es ihm angetan und er hofft, nach der Schule als Webdesigner tätig sein zu können.
Ronny Hendrichs	flashworker@hendrichs.de www.hendrichs.de	1977 geboren, Web-, Screen- und Multimediadesigner, QTVR-Produktionen (siehe www.qtvrs.de), Flash-Programmierer seit 1997 (z.B. www.Geraeusche.de), seit 09/2000 Software-Entwickler bei Vision42 AG (www.vision42.de)
Saban Ünlü	us@netTrek.de www.netTrek.de	Saban Ünlü, ein Freiberufler mit Schwerpunkt Flash, insbesondere Action Scripting.
Stephan Fischer	stephan@phaetons.de www.phaetons.de	Stephan Fischer programmiert Inter-/Intranetanwendungen mit Schwerpunkt Flash, CGI und Datenbanken.
Thomas Weller	thomas@weller-net.de www.weller-net.de	
Timo Meteling	timo.meteling@intabo.de www.intabo.de	Timo Meteling ist Geschäftsführer der Intabo Datendienstleistungs GmbH und sieht sich verantwortlich für den Einsatz von speziellen Flash-Lösungen.
Wojciech J. Kalka	w@kalka.org www.kalka.org	

Die CD-ROM

Zu jedem guten Lernbuch gehört eine CD mit Beispielen, Demos und begleitenden Materialien.

Die CD sollte man sich beim Lesen bereit halten. Zwar haben wir versucht, die jeweiligen ActionScript-Zeilen genau zu beschreiben, den Projekt-Zusammenhang kann man aber meistens in einem Buch nicht so gut begreifen wie in der entsprechenden Originaldatei.

Natürlich haben wir auch diverse Demo-Versionen von vorgestellten Programmen mit auf die CD gepackt, damit Sie sich selbst ein Bild machen können. Diese finden Sie im Verzeichnis Utilities. Im Verzeichnis Demoversionen haben wir mit der freundlichen Genehmigung von Macromedia Demoversionen von Fireworks, Dreamweaver und Dreamweaver UltraDev bereitgelegt. Allerdings waren zum Zeitpunkt der Drucklegung des Buchs die topaktuellen Demos zur Version 4 noch nicht zu erhalten. Diese sollten Sie sich bei Bedarf von der Macromedia-Seite www.macromedia.com/de herunterladen.

Die Verzeichnisse im Ordner Beispieldateien entsprechen den Überschriften der verschiedenen Kapitel, so dass Sie z. B. die Dateien des Kapitels »Flash und RealVideo« (im Teil Zusammenspiel) auf der CD im Ordner Zusammenspiel/Flash und RealVideo wiederfinden. Hier gibt es dann noch eine interne Hierarchie wie in diesem Fall die beiden Ordner Software und Übungen.

Im Verzeichnis Spiele befinden sich zwei Ordner »PathFindingEngine« und »DynamikMapGenerator«. Diese Dateien gehören zwar zu keinem Kapitel im Buch, sollten aber trotzdem keinem Spieleprogrammierer fehlen. Martin Fleck von www.action-script.com hat sie entwickelt und für die CD zur Verfügung gestellt.

Bitte beachten Sie unbedingt auch die Webseite zum Buch unter www.galileo-press.de, auf der wir regelmäßig Aktualisierungen zur Verfügung stellen werden.

Dann bleibt uns nur noch, Ihnen viel Spaß beim Lernen, Probieren und Flashen zu wünschen.

Index

Symbols

__proto__ 73
_parent 63
_root 63

Numerics

2D-Vektorisierung 271
3 110, 326
3D 264
  erstellen 265
  erstellen durch Sequenzen 267
  erstellen mit Pseudos 266
  FreeHand 272
  Handtracing 269
  Probleme 264
  Renderstyles 273
  Streamline 272
3D-Clipping 278
3D-Erstellung 265, 267
  mathematische 266
3D-Export 269
3D-Vektorisierung 272
3D-Werkzeug 267

A

Abnahme 30
Action Script Editor 432
Action Script Viewer 1.5 435
ActionScript 42
  Beispiele 86
  Grundlagen 46
  ins Aktionsfenster kopieren 430
  von Flash 4 zu Flash 5 43
Adobe Streamline 368
Aist
  Movie-Explorer 370
Aktionen
  in anderem Flashfile ausführen 377
Algorithmus 590
Alphaeffekte 36
Alpha-Tweenings 303
Analysephase 21
Anforderungsanalyse 21
Angebot 26
Animation
  mit Typographie 224
  Optimierung 148
Animationstools 430
AnySaver 433
Array-Objekt 54
Array-Zugriffsoperator 67
ASP 492
  E-Mail-Versand 496
  Formular versenden 496
ASP-Seite
  Erstellen 493
Audio- und Flash-Movies 355
Audiodateien
  Wiedergabe 348
Audioinformationen 240
Audiostream 348
Austastlücken 297
Auszeichnungsarten
  bold 215
  Farbe 216
  fett 215
  kursiv 214
Autolaunch 432
AVI 341

B

Bandbreitenengpässe 303
Bandwith Tuner 353
  Quellen 355
Behaviors 399
Behaviors-Inspektor 400
Benutzeranalyse 21
Bewegungstween 35
Bézierkurven 80
Bibliothek
  Aufbau 19
Bibliothek, interne
  ergänzen 646
Bild
  mit transparenten Verläufen importieren 648
Bildschirmschoner 432
Bildwiederholrate 111
Bitbull 435
Bitmap
  Nachbearbeitung 365
Blur-Effekt 193, 331
boolesche Variablen 84
Breakout 110
Browser
  Breite des Pop-up-Fensters 391
  rütteln 651
BrowserCache 654

Browserfenster
schließen 391
zusätzlich öffnen 388
Browser-Skripts 416
Browserunterstützung 389
Buchstaben
Laufweite 226
Buddy Saver 433
Bugliste 651
Bühne 15, 317
Bühnengröße 15

C

Cakewalk Pro Audio 253
CGI 503
Entwicklung 507
Erklärung 503
Installation auf Server 507
Konfiguration 509
mit verschiedenen Funktionen aufrufen 506
Chatroom
in Flash 422
CINOAN 433
Cookie 622
anlegen 622
auslesen 623
gespeicherten Wert ermitteln 623
Wert einstellen 417
Wert setzen 622
CorelDraw
Import 334
Count-down 178
CPU 297
Cubase VST 253

D

date 567
Dateigröße 34
Datenbank
Daten mit Generator auslesen 447
Datenquellen 443
Zeichen verwenden 448
Date-Objekt 52
Datum
einlesen 530
mit Flash 5 auslesen 567
Perl 558
PHP 565
Datumsfunktionen 567
div 84
Document Object Model 519
DOM 519
Dot-Syntax 62
DO-WHILE-Schleife 50
Dreamweaver 326, 397
Asset-Manager 328
freie Schriftauswahl 326
Objekt für Flash-Buttons 327
Steuerung von Flashfiles 381
Textblöcke als SWF generieren 327
Dreamweaver 4
Neue Objekte 326
Objekt für Flash-Text 326
Dreamweaver Ultradev 493
Drop Shadow 333
Drucken 618
als Bitmap 618
als Vektor 618
verhindern 619
duplicateMovieClip 92, 582
Dynamisches Menüsystem 640

E

Ebenen
Strukturierter Aufbau 18
Eckradius 75
Effekte 221
ELSE-Schleife 54
E-Mail
mit ASP versenden 496
mit vordefiniertem Subject senden 646
Emboss 333
Endlosschleife 50
Engine 279
Event 400
Event-Sounds 19
EWS88 MT 242
Exporteinstellungen 15
Extension Manager 399

F

Farbeffekt 36
Fenster
mit JavaScript öffnen 389
Film
starten 404
stoppen 404
zurückspulen 404
Filmsequenz 88
duplizieren 76
Position an Flash übergeben 385
positionieren 76
Fireworks 192, 329
Bitmap-Export 329
Effekte in Flash importieren 331
Live-Effekte 333
Fireworks-PNG
exportieren 330
Fireworks-Quelldateien
importieren 330
FLA
Aufbau 18
FlaBYTE Preloader 430
FlaPRINT 430
FlaSE 430
Flash
auf eine andere Datei zugreifen 426
aus Liste auswählen 405
Drucken 618
dynamisch generierte Inhalte einbinden 492
für TV produzieren 655
mit Skriptsprachen verbinden 136
runden 650
zoomen 405

Flash 4-Rotationsbug 650
Flash 5
JavaScript-Integration 396
Flash 5 Draw 94
Flash 5-Formulare 396
Flash CPU-Auslastung 355
Flash Foundry 434
Flash Frame
Gehe zu 402
Flash Player-Kontrolle 400
Flash Print 618
Flash Unprotect 430
Flash-Billard 130
Flash-Buttons einfügen 327
Flash-Chat 422
Features 424
Funktionsweise 423
Versionen 425
FlashDB 434
Flashdetection 604
Flashfile
Kommunikation über Frames 374
Flash-Film
bildschirmfüllend 381, 385
einziges Objekt 381
Fensterfüllend 384
Funktionalität erweitern 416
im gleichen Fenster 384
in einem Teil des Bildschirms 381
in einem Teil des Fensters 383
in neuem Fenster öffnen 381, 418
laden 403
öffnen im neuen Fenster 385
Schnelles Vorspulen 401
streamen 359
Streaming-Verhalten 354
Flashforge 5 433
Flash-Formular 298
Flash-Huhn 126
Flashjester Creator 433
Flashjester Entertainer 433
Flashjester JAVI 434
Flashjester JPrintor 434
Flashjester JTools 434
Flashjester Jugglor 434
Flash-Newsletter 524
Flash-Typografie 198
FlaWRITE 430
FlaWRITE.exe 128
Flax 435
FOR-IN-Schleife 49
Form2Flash 434
Formulare 298, 396
in Flash entwickeln 396
Formularprüfungen 406
Alphanumerische Prüfung 408
Datum 410
Eingabelänge 411
E-Mail 410
Fließkomma 412
Formatvorgabe 413
Ganzzahl 412
internationales Telefon 412
leere Eingabe 413
Listenauswahl 414
Postleitzahl 416
Prüfung Kreditkarten 409
Radio-Button 414
Sozialversicherungsnummer 414
Uhrzeit 414
URL 415
US-Telefon 415
Vergleichsprüfung 413
FOR-Schleife 48, 227
Framerate 111
FreeHand 195
Import 196, 334
FSCommand 375, 381
Fullscreen 652
Funktionsaufruf 374

G

Game Over 124
Gästebuch 298, 508
Gästebuch-CGI 504
Generator 438
Authoring Extensions 440
Datenformat 444
Developer Edition 463
Enterprise Edition 463
Offline-Generierung 439
Online-Generierung 439
Umgebungsvariable 443
Generator 2 434
Generatoren
Unterschiede 463
Generator-Inspektor 440
Generator-Objektpalette 440
Generator-Variable
ActionScript 442
Bühne 441
definieren 441
Definition 444
interne Definition 447
Wert für die Eigenschaft eines Platzhalters 442
GetURL 374, 380
GIFs
in Vektoren umwandeln 368
Go to Flash Frame 402
Goldwave 244
Grafik 34, 192
der Maus folgen lassen 634
in HTML-Seiten austauschen 419
Grafiksymbole 149

H

Handtracing 269
Highscore-Liste 136
  mit PHP 136
Hintergrund
  transparent 654
Hitfeld 150
hitTest 110
Hoolicon 435
How to 294
HTML-Datei
  dynamische 543
  über JavaScript erstellen 541
HTML-Pop-ups 298

I

Identifier 256
IF-Anweisung 46
  Verkettung 47
Illustrate 272, 274
Illustrator
  Import 334
Importformate 192
Importschutz
  von SWF-Dateien aufheben 430
Informationen
  filtern 29
ISIS 241

J

JavaScript 374, 532
  Datum auslesen 532
JavaScript Integration Kit 328, 396
  Verhalten 399
JavaScript-Variable
  aus Flash ändern 379
Jukebox
  Sound 256

K

Klassen 43
Kollision 130, 131
Kollisionenabfrage 134
Kollisionsabfrage 110, 119, 132
  Flash 4 134
  Flash 5 134
  quadratisch 135
  radial 134
Kommandozeileninterpreter 426, 427
  Erweiterung eines Flash-Movies 426
  selbst schreiben 429
  Zweck 427
Kommunikation
  mit der HTML-Seite 374
  zwischen JavaScript und Flash 375
Komprimierung 193
Konstruktor 71
Kontraste 218
Kopfhörer 243

L

Lade-Performance 297
Ladeverhalten 318
Ladevorgang
  bereits geladene Bytes anzeigen 430
Lautsprecherboxen 242
Lautstärke 259
Lesbarkeit 199, 211
Levels 62
Living Screen 433
Logic-Audio 253
Loops 248

M

Macromedia Exchange-Dienst 397
Macromedia Flash Dispatcher Behavior 420
Malprogramm 94
Maske
  mit Verlauf erstellen 648
Maskentexteffekt 233
Matrixrotation 283
Mausposition 86, 626
  ermitteln 626
Mausverfolgung 626
  Grafiken 634
  in Flash 5 630
  mit Objekten 628
Mauszeiger
  Objektverfolgung 629
Mediaplayer
  aus Flash steuern 651
Menü
  abhängige Skalierung 90
  horizontale Bewegung 89
Methoden
  definieren 70, 73
Microsoft Project 27
MIDI 251, 341
  aus Flash abspielen 655
MIDI-Audio-Studiolösungen 243
MIDI-Channel 252
MIDI-Datei 251
Ming 528
Ming-Bibliothek 528
Minislotmaschine 129
Modellierung 268
Monster Pack 237
Motionblur 194
Motionblur-Effekt 194
MotionTween
  optimieren 36
Movie
  rückwärts laufen lassen 647
MovieClip
  am rechten Rand 387
  positionieren 649
Movie-Explorer 370
MP3 239, 341
MP3-Dateien 237
Multimedia Xplorer 247
MySQL 486
MySQL-Datenbank
  Erstellung 486
  in PHP-Datei ausgeben 487
  Struktur 487

N

Navigation 319
new-Operator 70
Newsletter 524
  Administrierung 526

O

Objekt 51, 62
  Array-Objekt 54
  Date-Objekt 52
  in anderen Frames ansprechen 375
  neu in Flash 5 43
  selbst definieren 70, 146
  Selection-Objekt 51
Objektmanipulation 380
Objektorientierte Programmierung 62
OnionSkinMarker 37
Operator
  new 70
Optimierung 34
Outline
  verfolgen 647

P

Passwortabfrage 510, 513, 586
  anpassen 599
  mit Flash und CGI 595
  mit mehreren Usern 586
  Verschlüsselungstechnik 588
Perl 558
Pfade 62
Pfeiltasten 112
Pflichtenheft 25
Photoshop-Plug-ins
  in Fireworks 333
PHP 486, 565
PHP4-Skript
  Highscore 136
PHP-Datei
  Datenbankinhalte 487
PHP-Skript 139
Pixelbilder
  Import 15, 192
Play again 124
Plug-in
  Bezugsquellen 656
PNG-Datei 192
  exportieren 329
Poolbillard 130
Pop-up 388
  Fenster platzieren 220
  schließen 391
Portable Home Studio 254
Portastudio 255
Positionsermittlung 626
Preloaden
  von externen Movies 615
Preloader 138, 303, 610
  Alternativen 610
  Flash 2-kompatibel 610
  Flash 3-kompatibel 611
  Flash 4-kompatibel 612
  Flash 5-kompatibel 613
  mit JavaScript 613
Premiere 362
Programmierung
  objektorientiert 146
Projector Launcher 435
Projektion 279
Projektmanagement 20
Projektor
  von Cd starten 654
Projektorfenster
  Inhalt ausdrucken 430
Prototyp 73
Punktnotation 43
Punktzahl 163

Q

QuickTime 362
  zu Flash-Vektoren konvertieren 362
QuickTime Movie
  in einzelne Bitmaps umwandeln 363

R

RAD-Tools 370, 372
RealAudio-Dateien
  Standardeinstellungen 349
RealCodecs 356
RealNetworks 336
RealPlayer
  Goto-Befehl 357
  Inhalte entwickeln 342
  Kommandos 356
  Software 336
  Unterstützte Dateiformate 341
  unterstützte Formate 340
RealPlayer 8 339
RealProducer Basic 347
RealProducer Pro 347
RealServer Basic
  Betriebssysteme 345
RealVideo 336
  Bandwith Tuner 353
  LoadMovie-Kommandos 358
RealVideo-Daten
  live im Internet streamen 352
RealVideo-Film
  Auftruf 342
  Erstellung 346
  in das aktuelle Fenster laden 343
Reservierung 298
Rollover
  tweenen 319
Rotation 282
Rotationsbug 650
Rückwärtsspielen 647
runden 650

S

Sakkaden 213
Sampling-Auflösung 239
Sampling-Rate 238
Satz des Pythagoras 131
Schleife 48
  Do-While-Schleife 50
  ELSE-Schleife 54
  For-In-Schleife 49
  FOR-Schleife 48
  While-Schleife 50
Schreibmaschineneffekt 231

Schriften
_sans 208
_serif 208
Arial 201, 202, 203, 204, 205, 206, 207, 208, 209, 210, 211
aufgehellt 211
Aufmerksamkeit erzeugen 219
Avenir 204
Bodoni 202
bold 209
Centennial 202
Egyptienne 203
einbetten 218
Erkennbarkeit 199
fett 209
Frutiger 205
Futura 204
Garamond 202
gesperrt 210
Gill sans 205
Großbuchstaben 209
Hi Score 207
Hi WebT 207
Kontraste 218
Kuenstler Script 206
kursiv 210
Lesbarkeit 199
Modifikationen 209
negativ 211
Syntax 206
Times New Roman 201
unterschnitten 210
Verdana 203
Schriftwahl 216
Screendesign übernehmen 194
Screensaver Wizard 433
ScreenTime for Flash 433
Screenweaver 1 433
Scrollbalken 648
Scroll-Bildleiste 309
Scrollen
stoppen 157
Selection-Objekt 51
Sequenz
alle Bilder importieren 368
ShapeTween
optimieren 35
Shared Libraries 38
ShelExec.exe 428
Shopsystem 308
Sitecheck 294
Elsa-Webseite 311
Typographischer 290
www.die-waescherei.com 297
www.flashspiele.de 300
www.montblanc.com 316
www.schech.net 303
www.takito.com 294
www.toferer.at 307
Small Caps 213
Smart-Filmsequenz
Funktionsweise 100
SMIL 337, 350
Schnelleinstieg 350
SMIL-Datei
Aufbau 351
SML-Dokument
Parsen 522
Sortierreihenfolge von Objekten 285
Sound 236
Aufbau 19
bearbeiten 240
Dateiformate 237
Hardwarekomponenten 240
im Netz 236
klicken 650
Lautstärke 247
Methoden zuweisen 256
Panorama 260
Quellen 236
Softwarekomponenten 243
Startwerte 257
Verfügbarkeit 236
Sound Forge 245
Soundbearbeitung 247
Soundcube 237
Sounddatei
Qualität 237
Soundkarte 241
SoundObject 256
erzeugen 257
Sounds
aufnehmen 240
Speeddetection 607
Spieleprogrammierung 126
Coco-Catch 146
Techniken 134
Spielzeit 163
Spielzeitanzeige
aktualisieren 178
Spurenzahl 254
Stacking 285
Starfield 92
Stereo 238
Stoppuhr 143
Storyboard 24
Streamfähigkeit
Optimierung 353
Streaming Sound
Buffer verändern 649
Streaming-Media 336
String
Großbuchstaben 60
Stringaddition 57
STRING-Manipulationen 57
Struktur 17
Studio 251
virtuelles 253
SWF
positionieren 649
SWF-Ausgabebibliothek 528
SWF-Datei
Version bestimmen 651
SwiffPEG 435
Swift 3D 276, 432
Swift-Generator 434, 462
auf dem Server installieren 481
Dateien offline erzeugen 480
Dateien online erzeugen 481
Daten aus Datenbanken 477
Daten aus Textdateien 479
Datenbankabfrage 478
Flash-Dateien auslesen 469
Funktionsweise 464
Globale Angaben 470

Kommentare 470
Objekte beeinflussen 462
Probleme 483
Referer-Liste 479
Template-Datei 465
Variablen in ActionScript-Parametern 467
Variablen in Vorlage einfügen 466
Variblen in Textelemente 467
Vorlage 465
Vorlage erzeugen 465
Swift-Inspector 435
Swift-Skript 468
Bilder austauschen 476
Filmsequenzen austauschen 475
Schriftarten definieren 472
Sound austauschen 475
Text modifizieren 473
Textfelder modifizieren 474
Variablen 471
Swifty Utilities 435
SwiSH 435
SWT-Datei 438
Symbol
Bewegung 80
referenzieren 440
sinnvoller Einsatz 34

T

Task-Analyse 21
Tastatur
Implementierung 110
Team 28
Tell Target 69
testen 31, 115
Text
animieren 225
Aufbereitung 199
Kontraste 218
Links hervorheben 216
Optische Achsen 219
Optische Mitte 220
Rand 213
Versalien 213
Zeilenabstand 212
Zeilenbreite 212
Textanimationen
erstellen 430
Textdateien
Inhalte löschen 510
online editieren 510
Texteffekte 221, 224, 231
als Skript 225
mit duplicateMovie 233
tweenen 224
Textfeld
animieren 648
Textfile
cachen verhindern 651
Texturfüllungen
erstellen 649
verwenden 649
Texturing 268
this 71, 114
Time-Stretching 250
Transaktionen 358
Turbine 4 434
Turtle Beach Montego II 241
TV-Karte 243
Typografie 198
Animationen erstellen 224
experimentelle 199
klassische 199

U

Uhrzeit
einlesen 530
Umgebungsvariablen 443
URL Action Editor 435

V

Variable
an CGI senden 504
an einen Frame übergeben 378
auf Laden warten 505
aus anderem Flashfile auslesen 378
aus CGI auslesen 505
aus Textdatei auslesen 503
div 84
Formate 443
now 54
nur bestimmte senden 504
per JavaScript auslesen 653
Vecta 3D 275
Vecta3d Stand-alone 432
Vecta3d-Max 432
Vektorprogramme 334
Vektorraum 279
Verdecker 322
Vererbung 74
Verhalten 399
Fast Forward Flash 401
Go to Flash Frame Based on Cookie 402
Grafik 34
Load Flash Movie 403
Play Flash 404
Rewind Flash 404
Set Flash by List 405
Stop Flash 404
Zoom Flash 405
Verhalten-Inspektor 400
Verschlüsselung
bitweise 592
Verschlüsselungstechnik 588
Version 434
Video-Editing-Programme 362
Videoquelle 338
Videosequenz
in Flash optimieren 368
Wiedergabe in Flash 370
Videostream 346
Videotrick 315
Vorbereitungen 14

W

Wahrnehmungspsychologie 213
Wave-Editoren 243

WaveLab 246
Wave-Render-Programme 253
Weichzeichner 331
Werte
über eine URL an Flash weitergeben 541
WHILE-Schleife 50
WildSWfX 430
with 69
With-Operator 69
Woof 435
Wurzelziehen 135

X

XML 516
Anwendungsgebiete 517
Document Object Model 519
Einführung 516
in Flash 517
Vorteil 516
XML-Socket 44
XML-Tag 519

Y

Yugop-Menü 86

Z

Z-Clipping 285
Zeilenabstand 212
in Flash einstellen 213
Zeilenbreite 212
Zeit
messen 143
Zeitleisten
Aufbau 19
Zeitplan 26
Zoom 405
Zwiebelschaleneffekt 37